



نام درس

مهندسی نرم افزار ۱

تهیه و گرد آوری : ایمان مختاری

بهار ۸۸



فصل ۱ – محصول

مقدمه

يك بني سآختمانې چيست؟

يك سآختمان محصولي از يك يا چند بنا است و شامل مصاح سآختمانې چون سيمان؛ گچ و ... است

نرم افزار كامپيوتري چيست؟

محصولي از يك يا چند مهندس نرم افزار است كه شامل برنامه ها و داده ها است

نرم افزار كامپيوتري را چه كساني استفاده مي كند؟

هر فردي ممكن است به صورت مستقيم يا غير مستقيم از آن استفاده كند.

علت اهميت نرم افزار؟

تأثير گسترده بر تمام جنبه هاي زندگي

مراحل ايجاد نرم افزار چگونه است؟

با بكارگيري فرآيندي خاص و طي كردن يك روند مشخص

آشنا شدن با ميزان نفوذ نرم افزار

تأثيري بسيار ژرف و عميق دارد و نمونه بارز آن ماجراي y2k در اواخر سال ۱۹۹۹ بود

نقش نرم افزار؟

خودش يك محصول است و براي بكارگيري سخت افزار استفاده مي شود

وسيله اي براي ساخت و تحويل محصول به حساب مي آيد.

نقش دو گانه اي دارد

- نرم افزار چه در يك تلفن سلولي قرار داشته باشد يا درون يك كامپيوتر بزرگ، يك **انتقال دهنده** اطلاعات محسوب مي گردد. يعني اطلاعاتي را **توليد** مي كند ، **سازمان** مي دهد، **تحويل** مي گيرد، **نمایش** يا انتقال مي دهد. اگر از آن به عنوان وسيله انتقال دهنده محصول استفاده شود، به صورت پايه و اساس كنترل كامپيوتر (سيستم عامل) ، تبادل اطلاعات (شبکه ها) و ايجاد و كنترل برنامه هاي ديگر(ابزارهاي نرم افزاري و محيط آن) استفاده خواهد شد.



نمایش یا انتقال + تحويل + سازماندهي + توليد ➡ انتقال دهنده

تاریخچه نرم افزار

بحران نرم افزار اولین بار بصورت رسمی در کنفرانسی در سال ۱۹۶۸ میلادی مطرح گردید . عوامل اصلی این بحران عبارتند از :

تاخیر در تولید و تحويل نرم افزار
پیشرفت سریع سخت افزار
افزایش پیچیدگی محصولات

هزینه بالای ایجاد نرم افزار
نگهداری پرهزینه نرم افزار
کیفیت پایین نرم افزار

برنامه نویسی منفرد
برنامه نویسی تیمی

انواع روشهای برنامه نویسی رایج

مشکلات برنامه نویسی منفرد طولانی شدن زمان پروژه-افزایش هزینه ها-مشکل خطایابی-مشکل در ارزیابی میزان پیشرفت

خصوصیات نرم افزار

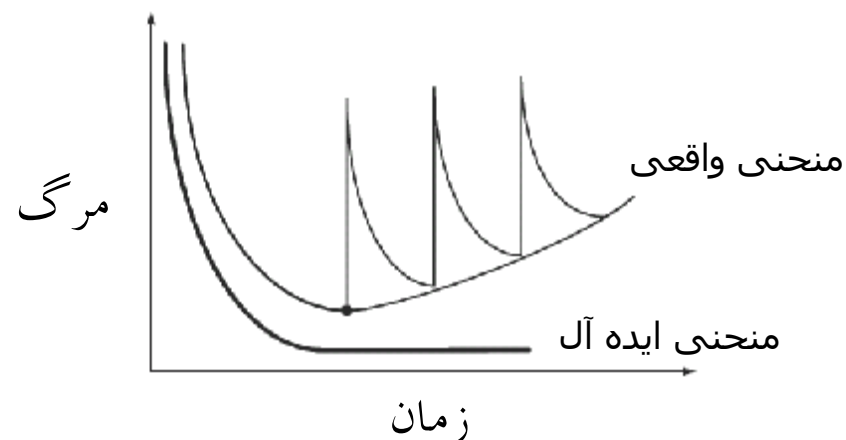
بهترین راه شناخت مشخصات نرم افزار ، **مقایسه آن با يك جزء ديگر** از محصولات ساخت انسان است (مثلا مقایسه با سخت افزار)

نتیجه مقایسه نرم افزار با سخت افزار

- ۱- نرم افزار توسعه و طراحی می شود، اما ساخته نمی شود (فیزیکی نیست)
- ۲- نرم افزار کهنه نمی شود ، اما زوال یافته و منسوخ می شود. (عدم حساسیت به عوامل محیطی)



نمودار نرخ خرابی سخت افزار



نمودار نرخ خرابی نرم افزار

- ۲- گرچه صنعت به سمت مونتاژ قطعات حرکت می کند ، اما نرم افزار سفارشی ساخته می شود و قابل استفاده فراوان است.
- يك قطعه نرم افزاري بايد طراحي و اجرا شود به گونه اي كه در بسياري از برنامه هاي مختلف ديگر استفاده شود.

کاربردهای نرم افزار

- ۱- اجرای رویه ای و از پیش تعریف شده (مگر در برنامه های هوش مصنوعی)
- ۲- محتوای اطلاعات از عوامل مهم در تعیین ماهیت برنامه نرم افزاري است

انواع نرم افزار ها:

- ۱- نرم افزار سیستم: سرویس دهی به سایر نرم افزار ها (مثل سیستم عامل)
- ۲- نرم افزار زمان واقعی (بلادرنگ) (RealTime): نظارت ، تحلیل و کنترل رویدادهای جهان واقعی (صنعت و...)
- ۳- نرم افزار تجاری: پردازش اطلاعات تجاری و بازرگانی (امور مالی و حسابداری)
- ۴- نرم افزار مهندسی و علمی: شامل الگوریتم های پردازش گر عددی (نرم افزارهای ستاره شناسی و...)
- ۵- نرم افزار جاسازی شده: جاسازی شده در دستگاههای ROM دار (ماشین لیاستشویی، ماکروفر و ...)
- ۶- نرم افزار کامپیوتر شخصی: نرم افزارهای کاربردی (مثل Word و ...)
- ۷- نرم افزار مبتنی بر وب: طبق پروتکل های اینترنتی و توسط مرورگرها بازیابی می شوند (سایت سازمان سنجش)
- ۸- نرم افزار هوش مصنوعی یا AI: حل مساله با الگوریتم های غیر عددی (شامل: تشخیص الگو/صدا/سیستم خبره و...)

انواع اشتباهات / تصورات و افسانه های نرم افزاری

- | | |
|---|-----------------------|
| وجود کتاب استاندارد و مراحل کار / آیا استفاده می شود؟
خرید کامپیوترهای جدید / استفاده از ابزارهای مهندسی نرم مهمتر است
اگر برنامه عقب باشد برنامه نویس اضافه می کنیم / عقبتر می افتیم | ۱. افسانه های مدیریتی |
| اهداف کلی برای شروع برنامه نویسی کافی است، جزئیات بعدا/اطلاعات باید دقیق باشد | ۲. افسانه های مشتری |
| زودتر کد را شروع کنیم کار تمام شود/زودتر کدنویسی شروع کنید دیرتر کار تمام می شود
وجود برنامه مشخص ساخت نرم افزار کافی است/برنامه مشخص عنصری از مراحل ساخت است | ۳. افسانه های متخصص |



فصل ۲ – فرآیند

مقدمه

برای تولید محصول نیاز به نقشه است مثلاً تولید یک برد الکترونیکی در تولید یک نقشه برای ایجاد یک محصول نرم افزاری چه کسانی در ارتباط هستند؟ مدیران - مهندسين - مشتریان

فرآیند نرم افزاری چیست؟ نقشه ای که با پیمودن آن نرم افزار تولید می شود.

دلیل اهمیت وجود فرایند در تولید نرم افزار؟

فراهم کننده پایداری - کنترل و سازماندهی فعالیت هاست

محصول یک فرایند نرم افزاری چیست؟ برنامه ها، اسناد و داده ها

نحوه اطمینان از صحت و درستی کار؟ توسط فرآیندها و مکانیزمای ارزیابی نرم افزار

مهندسی نرم افزار چیست؟

تعریف اول: استفاده از اصول ساده مهندسی به منظور رسیدن به یک نرم افزار مقرون به صرفه که قابل اطمینان بوده و روی دستگاههای واقعی کارآمد باشد.

تعریف دوم (IEEE):

- ۱- بکارگیری یک روش سیستماتیک، منظم کمیت پذیر برای توسعه/اجرا و پشتیبانی نرم افزار است یعنی بکارگیری مهندسی و شیوه های آن در نرم افزار.
- ۲- بررسی رهیافتهای آورده شده در قسمت (۱).



فرایندها، شیوه ها و ابزارها

مهندسی نرم افزار يك فناوري لایه به لایه است و نقشه (فرآیند) همانند چسبي است که لایه ها را به هم وصل می کند و اساس مهندسی نرم افزار لایه فرآیند است



گفتاش

لایه های مهندسی نرم افزار ←

حوزه های اصلی پردازش (KPAS) (نقاط کلیدی)

مبنای کنترل مدیریت پروژه های نرم افزاری

بستری که در آن روشهای فنی بکار گرفته می شود، محصولات تولید می شوند

محکهای بوجود می آیند، کیفیت سنجیده می شود و تغییرات احتمالی ترتیب داده می شوند

معرفی لایه روشها (شیوه ها): چگونگی انجام کار را از نظر فنی برای ساخت نرم افزار

معرفی لایه ابزارها: پشتیبانی از فرایندها و روش ها به صورت خودکار و نیمه خودکار توسط ابزارها

مهندسی نرم با کمک کامپیوتر (case) : استفاده ابزارها از اطلاعات سایر ابزارها

دیدگاه کلی از مهندسی در همه علوم : تحلیل خواسته ها ، طراحی ، ساخت ، تایید و مدیریت امور فنی

دیدگاه کلی از مهندسی نرم افزار : مرحله تعریف ، مرحله توسعه ، مرحله پشتیبانی



فازهای مهندسی نرم افزار

مرحله تعریف (What)

شناسایی نوع اطلاعاتی را که باید پردازش شوند، عملکرد مطلوب چه چیز است؟، نوع وضعیت سیستم مورد انتظار، نوع رابط هایی که باید ایجاد شوند. محدودیتهای موجود در طرح و معیارهای صحت را که برای تعریف یک سیستم موفق لازمند

مرحله توسعه (how)

چگونگی ساخته شدن داده ها، اجرای کار در ساختار نرم افزاری، چگونگی توصیف رابط ها چگونگی تبدیل طرح به زبان برنامه نویسی یا زبان غیر روبه ای) و چگونگی انجام آزمون را توصیف کند. (طراحی نرم افزار+ کد نویسی+ آزمون)

مرحله پشتیبانی (Change)

اصلاح خطا، ایجاد تطابقات لازم در حد لازم در رابطه با محیط نرم افزاری و تغییراتی ناشی از بهبود کار بخاطر تغییر نیاز مشتری
اصلاح (دید جدید) ، تطابق (تغییر شرایط محیط) ، بهبود وضعیت (افزایش امکانات)، پیشگیری (مهندسی مجدد و اصلاح آسانتر)

+فعالیت های چتری یا فعالیت های تکمیل کننده فازهای مهندسی نرم افزار:

- کنترل پیگیری پروژه نرم افزاری
- تضمین کیفی نرم افزار
- مدیریت طرح نرم افزاری
- آماده سازی اسناد و تولید
- مدیریت قابلیت استفاده مجدد
- بازنگری های فنی رسمی
- ارزیابی
- مدیریت خطر (ریسک ها)

آم کتاب

فرآیند نرم افزاری

بعضی از کارها که هر کدام مجموعه ای از کارهای مهندسی نرم افزاری، معیارهای پروژه محصول کار و نقاط تضمین کیفیت هستند، باعث می شود که این مجموعه های معین فعالیتی در مشخصه های پروژه نرم افزاری بکار گرفته شده و در مجموعه تقاضاهای تیم پروژه نیز باشند.

موسسه مهندسی نرم افزار (SEI) مدل جامعی ارائه داده که روی مجموعه ای از تواناییهای مهندسی نرم افزار که باید برای دسترسی سازمانها به سطوح مختلف رشد فرآیند وجود داشته باشد، پیش بینی شده است.

این طرح درجه بندی **میزان تطابق با توانایی مدل کامل (CMM) را مشخص** می کند که فعالیتهای اصلی لازم در سطوح مختلف فرآیند تکامل را تعریف می کند.

سطوح تکامل فرآیند (CMM)

- سطح ۱: **اولیه**: روش نامشخص و بستگی به تلاشهای انفرادی
- سطح ۲: **قابل تکرار**: جهت تعیین هزینه و زمان کارایی
- سطح ۳: **تعریف شده**: استفاده از فرآیند سازمانی مستند و تایید شده
- سطح ۴: **مدیریت شده**: اندازه گیری کامل فرآیند نرم افزار و کیفیت محصول.
- سطح ۵: **بهینه سازی**: اصلاحات با بازخوردها و ایده های نو

هر سطح بالای سطح ۲ زیر مجموعه ای از سطح قبلی است.

سطح ۴ دربرگیرنده مفهوم اندازه گیری و استفاده از متریک است.

• متریک یک کمیت معنادار است که می توان آن را از روی بعضی از اسناد یا کدی در داخل پروژه نرم افزاری استخراج نمود.

سطح ۵ بالاترین سطح قابل دسترسی است. این سطح نمایانگر آنالوگ مکانیزمهای کنترل کیفی نرم افزار است که در مکانیزمهای دیگر و صنایع کامل تر وجود دارند.

سطوح تکامل فرآیند (CMM)

SEI دارای حوزه های اصلی در فرآیند است (KPA) که هر کدام یک سطح رشد و تکامل دارند. KPAها آن گروه از فعالیت های مهندسی را توصیف می کنند که باید برای صورت گرفتن کار درست در سطح بخصوصی موجود داشته باشند.

هر KPA با شناسایی مشخصه های زیر توصیف می شود:

اهداف - اهداف کلی که KPA باید به آن دست یابد .

تعهدات - نیازمندیهای که باید مرتفع شود (در روند انجام فرایند)

توانائیهها - لوازم مورد نیاز برای رسیدن به تعهدات

فعالیتها - کارهای خاصی که لازمند تا به عملکرد KPA دست یابند

روشهای کنترل پیاده سازی - شیوه ای که با آن فعالیتها در جای خود قرار می گیرند کنترل می شوند.

روش هایی برای ارزیابی درستی پیاده سازی - شیوه ای که با آن عمل درست برای KPA مشخص می شوند

هجده KPA (هر کدام با استفاده از این خصوصیات توصیف شده اند) در طول فرآیند تکمیل کار تعریف شده و در سطح مختلف فرآیند تکاملی بصورت طرح درآمده اند . KPA زیر را در هر سطح تکمیلی فرآیند به دست می آوریم :

بازبینی های یکسان .
همکاری درونی گروهی .
مهندسی محصولی نرم افزاری .
مدیریت یکپارچه نرم افزاری .
برنامه آموزشی .
تعریف فرآیند سازمانی .
نقطه تمرکز فرآیند سازمانی

سطح ۳

مدیریت تغییر فرآیند
مدیریت تغییر تکنولوژی .
پیشگیری از نقیصه و اشکال

سطح ۵

مدیریت پیکربندی نرم افزار .
تضمین کیفیت نرم افزار .
مدیریت پیمان کاریهای نرم افزار.
پی گیری نظارت پروژه نرم افزاری .
مدیریت نیازمندیها.
برنامه ریزی پروژه

سطح ۲

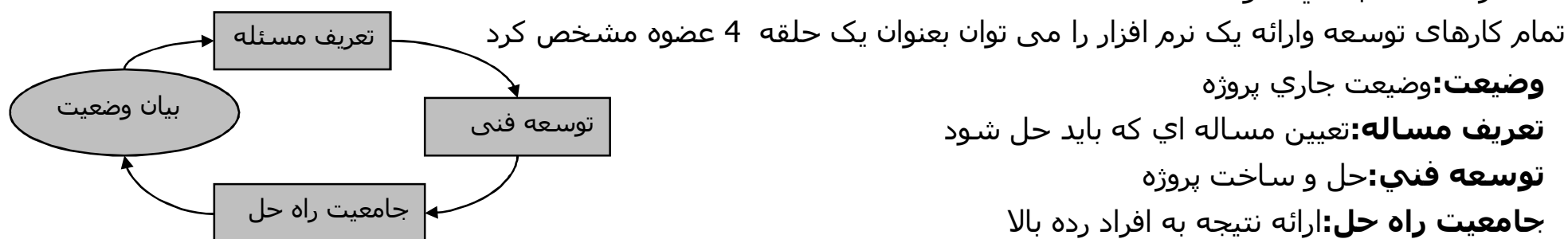
مدیریت کیفیت نرم افزار.
مدیریت فرآیند کمی.

سطح ۴

مدل فرآیند (Paradigm)

تیمی از مهندسين نرم افزار بايد استراتژي (راهبرد) ارائه دهد که شامل لایه های فرآیند (کفاش) و فازهای عمومی (تعریف/توسعه/حمایت) اغلب این راهبرد را مدل فرآیند یا پارادایم مهندسی نرم افزار می نامند

نکته: برای انتخاب مدل فرآیند براساس ماهیت پروژه و کاربرد آن روشها و ابزار مورد استفاده و کنترل ها و موارد قابل ارائه که لازمند انتخاب می شوند.

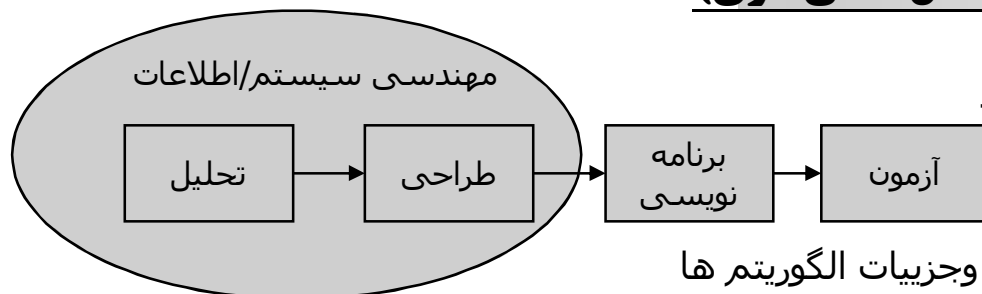


غیر از فاز بیان وضعیت سایر فازها می توانند خود دارای مراحل چهارگانه باشند

انواع مدل های فرآیند ارائه شده:

۱. مدل ترتیبی خطی
۲. مدل ایجاد نمونه اولیه (Prototyping Model)
۳. مدل RAD
۴. مدل افزایشی (Incremental Model)
۵. مدل حلزونی (Spiral Model)
۶. مدل حلزونی برنده - برنده (Win-Win)
۷. مدل توسعه همزمان
۸. مدل توسعه مبتنی بر مولفه (Component Base Development)
۹. مدل روشهای رسمی (Formal Method)
۱۰. تکنیکهای نسل چهارم

۱- مدل ترتیبی خطی (چرخه حیات کلاسیک) (مدل آبشاری) (مدل خطی سری)



مهندسی سیستم/اطلاعات: جمع آوری نیازها (سخت افزار/db) و امکان سنجی و تحلیل و طراحی سطح بالا

تحلیل: جمع آوری نیازهای نرم افزاری

طراحی: تمرکز طراحی بر ساختمان داده/معماری نرم افزار/رابطها و جزئیات الگوریتم ها

کد: برنامه نویسی و طراحی نرم افزار

آزمون و پشتیبانی: تایید صحت برنامه / رفع خطاهای احتمالی و تطابق با محیط

معایب روش ترتیبی خطی

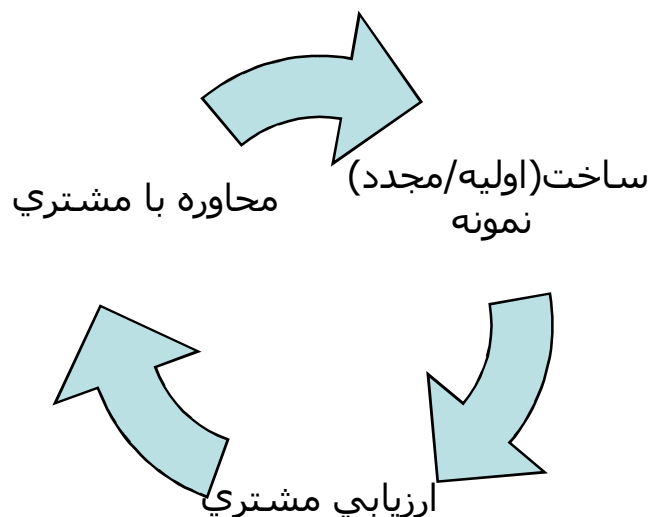
- توسط پروژه ممکن است موارد گنج کننده ای برای اعضا رخ دهد
- برای مشتری مشکل است که تمام نیازهای خود را به طور مشخص و یکجا بیان کند
- رونمایی آخرین مرحله کار و انتظار مشتری و احتمال ریسک (اشتباهات)

۲- مدل نمونه سازی (نمونه اولیه) (Prototype)

اگر اهداف کلی ذکر شود مناسب است

معایب روش نمونه اولیه

- عدم آگاهی مشتری از فوری بودن پروژه و عدم پیش بینی اهداف بلندمدت در آن
- به دلیل سرعت امکان استفاده از الگوریتم ناکارآمد



۳- مدل RAD (Rapid Application Development) = ساخت سریع برنامه ها

- زمانی که تاکید بر دوره کوتاه است این روش مناسب است
- نوعی مدل خطی سرعت بالاست که بر استفاده از ساخت بر مبنای قطعات اولیه استاساسا برای کارهای سیستمهای اطلاعاتی استفاده می شود .

فازهای مدل RAD

مدلسازی تجاری

مدلسازی جریان اطلاعات بین توابع تجاری

مدلسازی داده ای

تبدیل جریان اطلاعات فاز مدلسازی تجاری به مجموعه اشیای داده

مدلسازی فرآیند

تبدیل اشیای داده ای، طوریکه جریان اطلاعات لازم برای پیاده سازی توابع تجاری حاصل شود

ایجاد کاربرد

به جای تولید نرم افزار با استفاده از زبانهای متعارف برنامه نویسی متعارف نسل سوم فرآیند RAD تلاش می کند تا از جزء های برنامه موجود مجددا استفاده کرده یا جزء های قابل استفاده مجددی را تولید کند. (نسل چهارم)

آزمون و گردش (خاتمه)

به دلیل آزموده شدن قبلی دیگر دوباره آزموده نمی شوند.

معایب روش RAD

- برای پروژه های بزرگ نیازمند تعداد زیادی متخصص (وجود چند تیم)
- توافق دوطرف (مشتری/متخصص) بر انجام سریع پروژه
- اگر کارایی مدنظر باشد RAD مناسب نیست
- زمانیکه خطرات فنی وجود دارد چندان مناسب نیست.



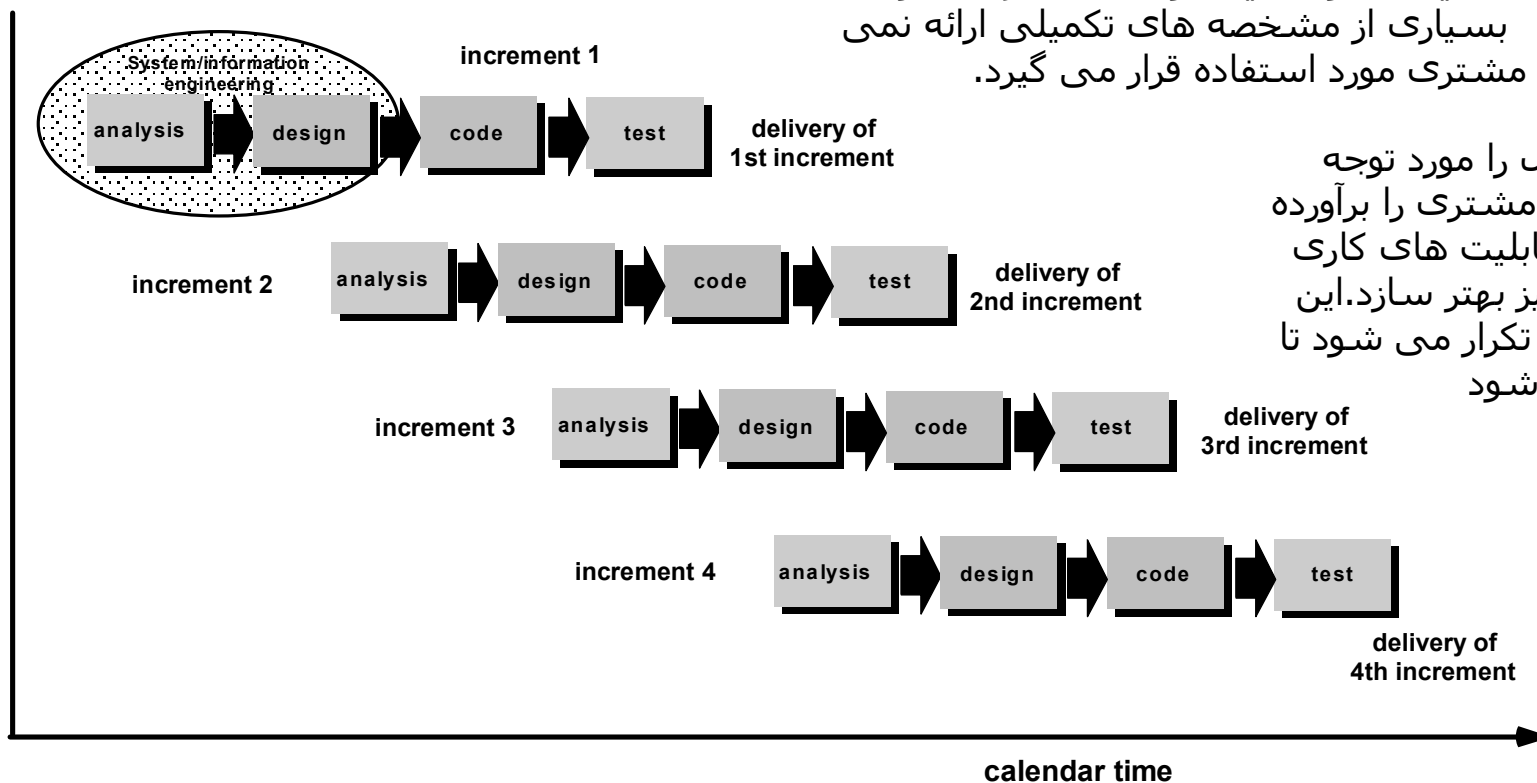
مدلهای فرآیند تکامل نرم افزار (تکاملی یعنی خروجی یک سیکل ورودی سیکل بعدی)

در این مدلها تکرار اصل مهم تکامل است

۴- مدل افزایشی

همان مدل خطی با دیدگاه تکرار است و برخلاف نمونه سازی، خروجی هر مرحله دور ریخته نمی شود و برای تکامل در چرخه باقی میماند

اولین تکرار سبب ایجاد هسته اصلی محصول می شود یعنی نیازهای اولیه مورد توجه قرار می گیرند اما بسیاری از مشخصه های تکمیلی ارائه نمی گردند. هسته محصول بوسیله مشتری مورد استفاده قرار می گیرد.

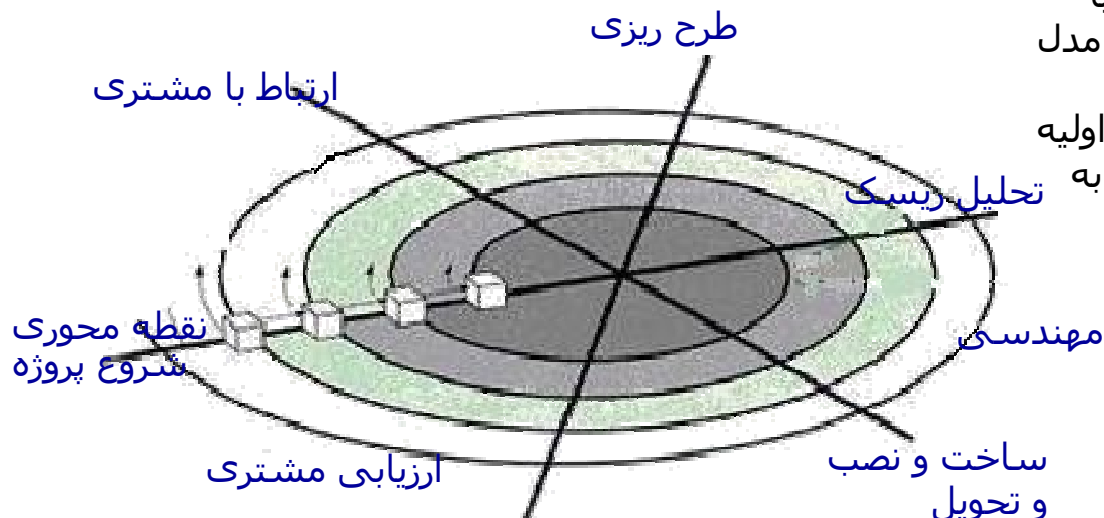


این طرح اصلاح هسته محصول را مورد توجه قرار می دهد تا بهتر نیازهای مشتری را برآورده نموده و ارائه مشخصه ها و قابلیت های کاری اضافی مورد نظر مشتری را نیز بهتر سازد. این فرآیند بدنبال تحویل هر بخش تکرار می شود تا وقتی که محصول کامل تولید شود

زمانی مفید است که حمایت لازم برای پیاده سازی کامل وجود ندارد

۵- مدل مارپیچی Spiral

- توسط بوهم پیشنهاد شد ، ماهیت تکرار نمونه سازی با محاوره مشتری را با جنبه های کنترلی و سیستماتیک مدل خطی ترکیب نموده است
- پروژه در چندین مرحله ساخته می شود ، شاید مراحل اولیه تنها روی کاغذ یا الگوی اولیه مهندسی باشد و سپس به ایجاد/ارتقا و نگهداری محصول برسیم
- یک دور مارپیچ شامل ۳ تا ۶ فاز کاری است



پروژه های نگهداری محصولات	
پروژه های بهینه سازی محصولات	
پروژه های ساخت محصول جدید	
پروژه های توسعه مفهوم	

- ارتباط با مشتری: بررسی نیازها
- برنامه ریزی (Planing): تعریف منابع و زمانبندی
- تحلیل خطر: مشخص کردن ریسکها و خطرات
- مهندسی: طراحی و ایجاد برنامه
- ساخت و ارائه: ساخت/آزمون/آموزش/نصب
- ارزیابی مشتری: بدست آوردن واکنش مشتری و بازخورد

تکرار فراوان از بروز خطا جلوگیری می کند

معایب روش مارپیچی

- قانع کردن مشتری در دسرساز است
- ممکن است اطلاعات نادرست به ما بدهند

۶- مدل ماریچی Win-Win

- به دلیل امکان قرار نگرفتن اطلاعات کافی در ارتباطات با مشتری ، دنبال ارتباط با افراد حائز شرایط برنده می رویم

تفاوت ماریچی با ماریچی Win-win :

در مدل حلزونی بجای یک رابطه ارتباط مشتری ، فعالیت های زیر تعریف می شوند:

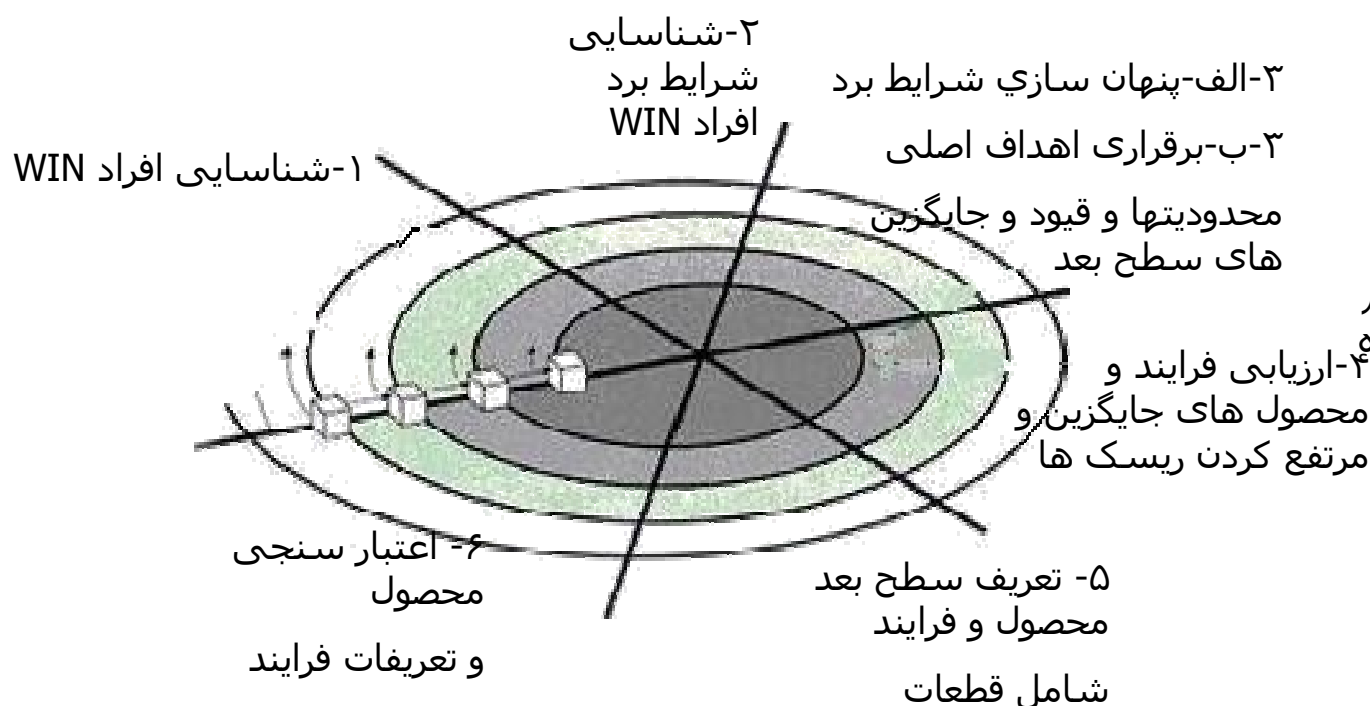
- ۱- شناسایی سیستم یا دارندگان اصلی سیستم های فرعی
- ۲- تعیین شرایط برد دارندگان سهم.
- ۳- مذاکره در مورد شرایط برد سهامدار برای تطابق دادن آنها در مجموعه ای از شرایط win-win برای همه افراد درگیر.

نقاط اطمینان (تکیه):

معیاری برای اندازه گیری پیشرفت

دیدگاههای نقاط تکیه (معیار پیشرفت):

اهداف دوره زندگی (lco) : اهداف فعالیتها
معماری دوره زندگی (lca) : اهداف سیستم
توانایی عملیات اولیه (loc) : آماده سازی پیاده سازی



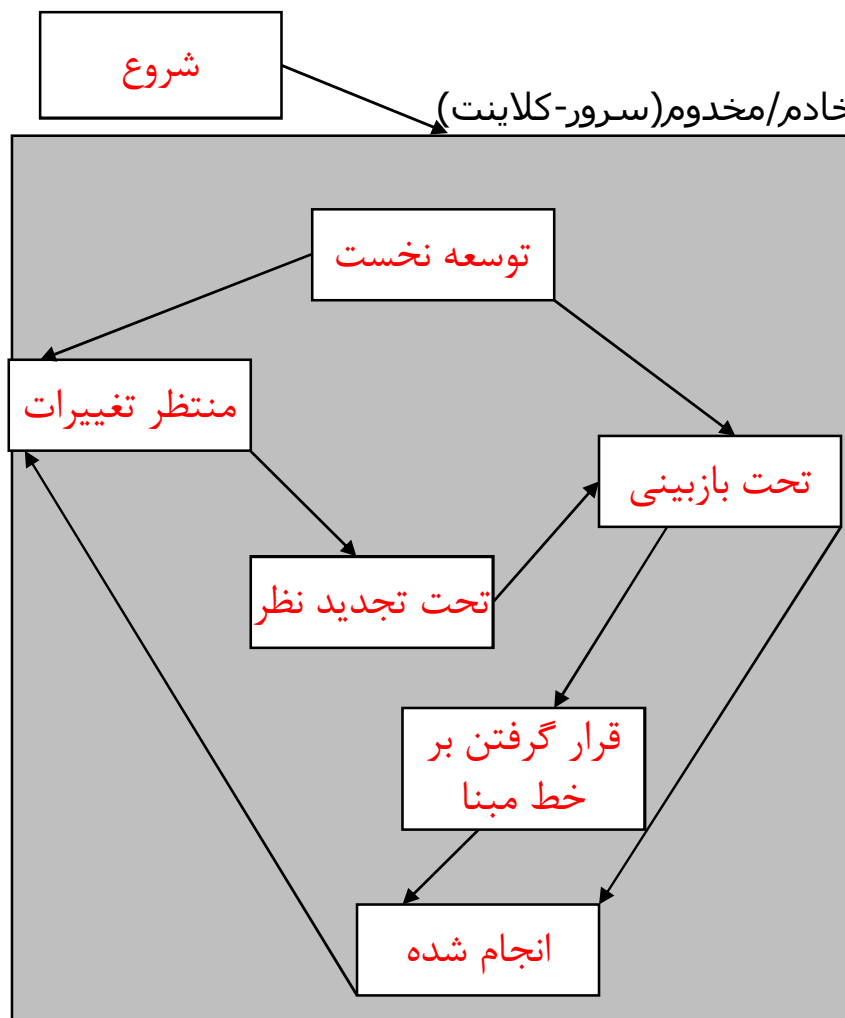
۷- مدل توسعه همروند (همزمان) (مهندسی همزمانی)

- برای دنبال کردن وضعیت يك پروژه
- تمام فعاليتها به صورت همزمان نمایش داده مي شوند و حاليهاي فراواني ايجاد مي شود كه از يك حالت به حالت ديگر با رخ دادن يك رویداد مي رويم
- تقسيم نيروها و حركت موازي
- مدل توسعه همروند ابزاري مناسب براي توسعه و ايجاد برنامه هاي خادم/مخدوم (سرور-كلاينت)
- ابعاد سيستمهاي كلاينت/سرور
- ۱. بعد سيستم: ساخت/مونتاژ و استفاده
- ۲. بعد مولفه: طراحي /بررسي

نحوه همزمانی در توسعه همروند:

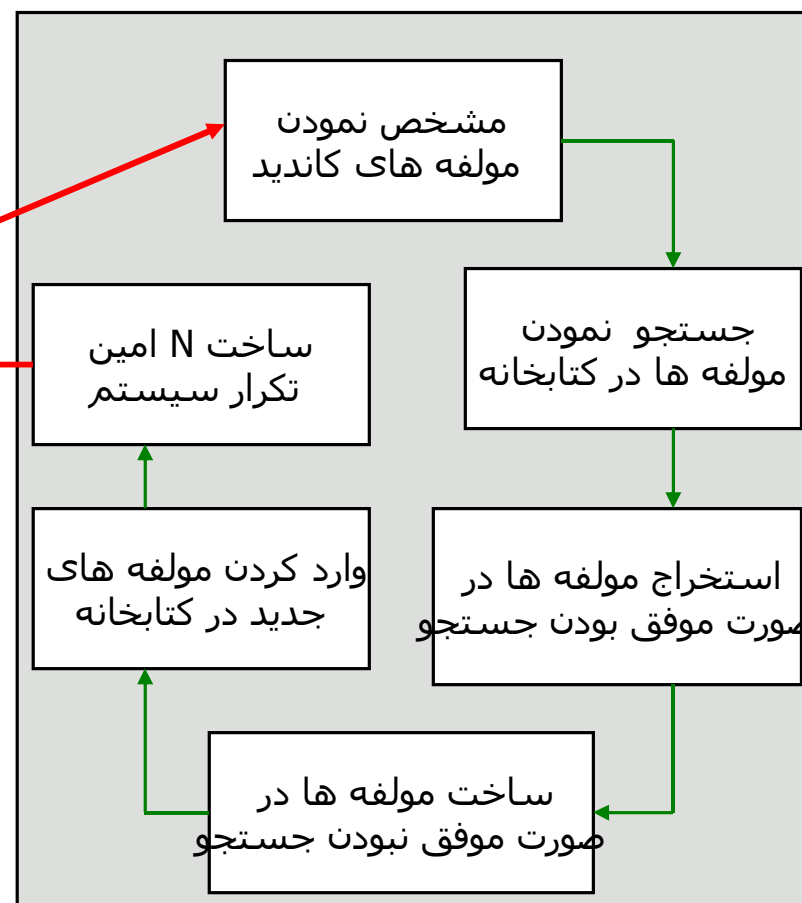
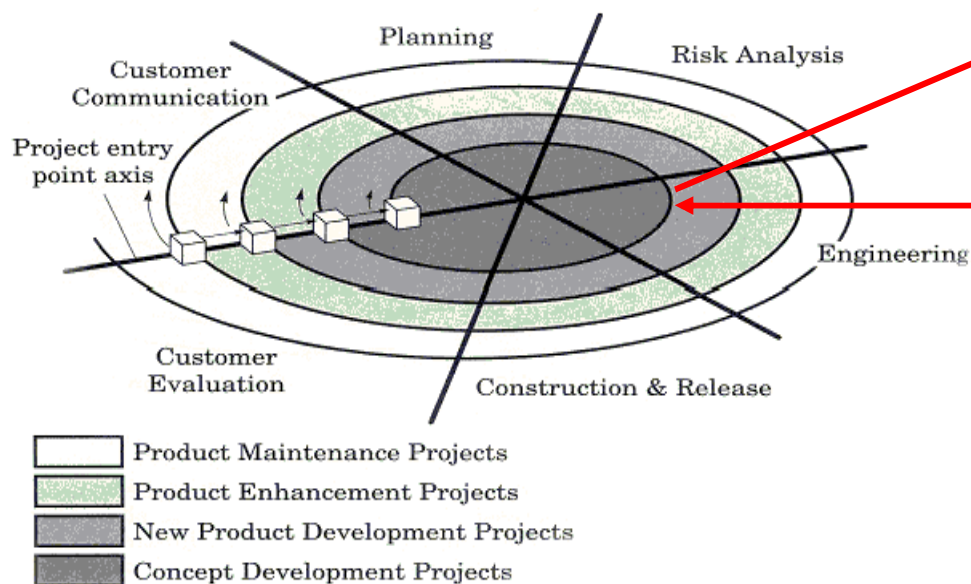
۱- همزمانی فعاليتهاي سيستم و مولفه

۲- طراحي همزمان اجزاي موجود در برنامه كلاينت / سرور



۸- توسعه مبتنی بر اجزا (مولفه) (CBD) Component Base Development

- دارای ماهیت تکاملی است و بسیاری از خصوصیات مدل حلزونی را دارد
 - در کنار ویژگیهای مدل حلزونی از تکنولوژی شی گرای و object نیز استفاده می کند
 - به سبب تاکید بر استفاده مجدد و اشیا، کاهش هزینه و افزایش سرعت را به دنبال دارد
- UML زبان مدلسازی است که نماینده این مدل در صنعت محسوب می شود



۹- مدل روشهای رسمی

- تعیین رسمی مشخصه های نرم افزار با استفاده از روابط ریاضی
- ابهام کمتری نسبت به روشهای غیر قراردادی دارد
- وجود نرم افزاری بدون مشکل را تضمین می کند

معایب روشهای رسمی

- زمان و هزینه بالا
- آموزش گسترده جهت مهندسی نرم افزار
- دشواری بکارگیری مدلها در تعامل با مشتریانی که فاقد دید فنی می

۱۰- تکنیکها و فنون نسل چهارم (4GT)

- محور اصلی این روش، استفاده از ابزارهای مهندسی نرم افزار است
- ابتدا يك مشخصه از نرم افزار را در سطح بالا (نماد و مدل) طراحی و سپس کد تولید می شود

خصوصیات محیطهای توسعه نرم افزار نسل چهارم:

- ۱- زبان غیر رویه ای (4GL) ۲- ویرایش داده ها ۳- فرمها و ارتباطات متقابل ۴- گرافیک بالا
- ۵- امکان تولید کد و برنامه ۶- توانایی تولید صفحه گسترده و HTML

- استفاده از 4GT به همراه ابزارهای مهندسی نرم، راه مناسب برای بسیاری از مسائل نرم افزاری
- کاهش زمانبندی تولید نرم افزار و میزان تحلیل پروژه های کوچک
- نیاز به تحلیل/طراحی و آزمایش بیشتر برای پروژه های بزرگ

ادعای موافقین

وضعیت فعلی روشهای 4GT

ادعای مخالفین 4GT

خیلی ساده تر از زبانهای برنامه نویسی نیست- کد تولید شده چندان کارآمد نیست- امکان تولید سیستم بزرگ نیست

رابطه محصول و فرآیند

اگر فرایند ضعیف باشد محصول نهایی بدون شك با مشکل مواجه می شود

مثل مهندسی نرم افزار اتاق پاك
طراحی سیستم با مدلهاي آماری قبل از
نوشتن کدهای برنامه و استفاده از مدل
Incremental به صورت زیر سیستمهاي
سیستم



بخش دوم - مدیریت پروژه های نرم افزاری

فصل سوم - مفاهیم مدیریت پروژه

مقدمه

• تعریف مدیریت پروژه (Pressman) :

مدیریت پروژه شامل برنامه ریزی ، نظارت و کنترل افراد ، فرایند و رخدادهایی می باشد که در طول تکامل نرم افزار از مفهومی اولیه به یک پیاده سازی عملیاتی منجر می گردد.

مهندس نرم افزار : فعالیت های روزمره طرح ریزی و کنترل کار های فنی خود را اداره می کند.

مدیران ارشد : ارتباط بین امور و متخصصین نرم افزار را فراهم می کند.

مدیران پروژه : گروه یا تیم مهندسین نرم افزار را کنترل و طرح ریزی میکند

و.....

• سطوح
مدیریت

• دلیل اهمیت مدیریت پروژه

ایجاد نرم افزار کامپیوتر فعالیت پیچیده است ، بخصوص اگر افراد متعددی برای زمان طولانی در آن کار نمایند ، دلیل اصلی نیاز به مدیریت پروژه این موضوع است.

• محصول مدیریت پروژه؟

طرح پروژه با آغاز و شروع فعالیت های مدیریت تهیه و تولید می شود. این طرح روند و کارها یی که باید انجام شود ، افرادی که کارها را انجام می دهند و مکانیزم ارزیابی خطرات ، کنترل تغییر و ارزیابی کیفیت را تشریح می کند.

• اطمینان از صحت مدیریت

هرگز به طور قطعی نمی توانید اطمینان حاصل کنید که طرح پروژه درست است مگر آنکه محصول یا کیفیت را در زمان مقرر و یا بودجه تعیین شده ارائه کنید.

مولفه های مدیریت پروژه نرم افزاری (4p)

Process	فرایند	People	افراد
Project	پروژه	Product	محصول

مولفه های مدیریت پروژه نرم افزاری (4p)

۱- افراد

پرورش پرسنل نرم افزار کاملاً ماهر و با انگیزه در واقع فاکتور انسانی { افراد یا پرسنل } آنقدر مهم است که مو سسه مهندسی نرم افزار يك مدل تكامل توانایی مدیریت نیروی انسانی به نام PM-CMM را ارائه داده است.

مدل رشد مدیریت افراد (PM-CMM) عملکرد های کلیدی : استخدام ، مدیریت عملکرد، جبران آموزش و پیشرفت شغلی و طراحی سازمان و توسعه تیمی را تعریف می کند

۲- محصول

قبل از طرح ریزی پروژه باید اهداف و حیطه محصول مشخص شده و محدودیت های فنی و مدیریتی تشریح شوند. بدون این اطلاعات نمی توان برآوردهای معقولی از هزینه ، ارزیابی موثر خدمات ، تجزیه و تحلیل کارها و وظایف پروژه یا برنامۀ زمان بندی قابل اجرا پروژه همراه با نشانه های درست پیشرفت را ارائه کرده است. در بسیاری از موارد این فعالیت بخشی از مهندسی سیستم یا مهندسی روند کار شروع میشود و به عنوان مرحله اول در تحلیل نیازمندی های نرم افزار ادا می پیدا می کند

۳- فرآیند

پروسه چهارچوبی را ارائه می کند که به کمک آن می توان طرح جامع توسعه نرم افزار را تهیه کرد . مجموعه های مختلف کاری - وظایف ، مراحل برجسته کاری ، محصولات کاری و مقاطع تضمین کیفیت - به فعالیت های اصلی این امکان را می دهد تا در مشخصه های پروژه نرم افزار و نیازمندی های تیم پروژه به کار گرفته شوند. در نهایت فعالیت های حفاظتی مثل تضمین کیفیت نرم افزار، مدیریت پیکر بندی نرم افزار و اندازه گیری و مدل فرایند حاکم هستند

۴- پروژه

ما پروژه های نرم افزاری کنترل و طرح ریزی شده را به یک دلیل اصلی انجام می دهیم زیرا تنها راه شناخته شده برای اداره کردن و کنترل پیچیدگی است و هنوز هم با آن در کشمکش و جدال هستیم . نرخ موفقیت پروژه های نرم افزاری تا حدی بهبود پیدا کرده است اما نرخ شکست و عدم موفقیت پروژه هم چنان بالا تراز حدی است که باید باشد . برای پرهیز از شکست پروژه ، مدیر پروژه نرم افزار و مهندسین نرم افزاری که محصول را می سازند باید از علائم اخطار دهنده پرهیز کنند ، عوامل موفقیت را که به مدیریت پروژه خوب می انجامد درک کنند و روش مشترک و معقولی برای طرح ریزی ، نظارت و کنترل پروژه ارائه کنند.

افراد

اهمیت افراد

- در تحقیق انتشار یا فته IEEE از ۳ شرکت فنی بزرگ خواسته شد تا مهم ترین عامل دخیل در پروژه نرم افزار موفق را ذکر کنند . پاسخ آنها به شرح زیر است :
- ۱- آن چیز انسان مورد استفاده ما نیست بلکه افراد یا پرسنل است .
 - ۲- مهمترین عاملی که باعث موفقیت این پروژه شد برخورداری از پرسنل با هوش بود... دیگر عوامل اهمیت کمتری دارند.
 - ۳- تنها قاعده ای که در مدیریت از آن استفاده می کنیم تضمین در اختیار داشتن افراد خوب و ایجاد محیطی است که در آن بتوان افراد خوبی درست کرد.
- مدیران میگویند که پرسنل عامل اصلی هستند اما گاه کردار و عملکرد آنها با گفتارشان یکی نیست .

انواع گروههای افراد در فرایند ساخت نرم افزار

- ۱- **مدیران ارشد** که موضوعات و مسائل کاری را مشخص میکنند که اغلب تاثیر زیادی بر پروژه دارند (تنظیم قرارداد-....).
 - ۲- **مدیران پروژه (فنی)** که باید افرادی را که کار نرم افزار انجام میدهند برنامه ریزی، تخصیص وظایف، سازماندهی و کنترل کنند.
 - ۳- **مجریان (متخصصین)**: افرادی که مهارت های فردی مورد نیاز جهت مهندسی محصول یا کاربرد ارائه میکنند.
 - ۴- **مشتریان** کسانی که نیازمندی های نرم افزار مورد نیاز را مشخص میکنند .
 - ۵- **کاربران نهایی** که با عرضه نرم افزار جهت استفا ده از آن بهره میگیرند.
- تیم پروژه برای کا رآمد بودن با ید به نحوی سازماندهی شود که مهارت ها و قابلیت های هر فرد را به بیشترین حد ممکن برساند و این وظیفه رهبر تیم است.

* رهبران تیم

مدیریت پروژه یا رهبر تیم فعالیتی در تعامل با افراد است و به این دلیل مجریان و متخصصین ذیربط ، اغلب رهبران تیم خوبی نیستند . آنان از مجموعه مهارت های فردی درست و لازم برخوردار نیستند .

مدل MOI (برای رهبری) :

انگیزه : قابلیت ترغیب پرسنل فنی در ایجاد بهترین قابلیت ها.

سازماندهی : توانایی ایجاد و برقراری پروسه های موجود که امکان تبدیل مفهوم اولیه به محصول نهایی را فراهم می کند.

ایده ها یا ابداع : توانایی ترغیب افراد به خلاقیت به هنگام کار در محدوده تعیین شده برای محصول .

چهارویژگی مدیر پروژه کارآمد

حلهو موسا



۱. **حل مسئله** : مدیر پروژه نرم افزار کارآمد میتواند مسائل فنی و سازمانی شایع را تشخیص دهد و راه حل اصولی را سازماندهی و مطالب اندوخته شده از پروژه های قبلی را در موقعیت های جدید اعمال کند و در صورت بی ثمر بودن راه حل های اولیه برای تغییر جهت انعطاف داشته باشد.
۲. **هویت مدیریتی** : مدیر پروژه خوب باید مسئولیت پروژه را بر عهده بگیرد: او باید برای بدست گرفتن کنترل در زمان مقتضی و محاز کردن پرسنل فنی به پیگیری غرایز خود ، از اتکا به نفس خوبی برخوردار باشد.
۳. **موفقیت**: برای بهینه کردن بهروری تیم پروژه مدیر باید موفقیت و نوآوری را تشویق کند و با اعمال خود نشان دهد که اگر افرادی به طور کنترل شده رسیک کنند با محازات روبرو نمی شوند.
۴. **ساخت تیم و تاثیر گذاری**: یک مدیر موفق و موثر با ید مردم شناس باشد و تحت شرایط فشار بتواند اوضاع را تحت کنترل در آورد.

تیم نرم افزار

تعداد ساختارهای سازمانی انسانی برای توسعه نرم افزار، به اندازه تعداد سازمانهای توسعه دهنده نرم افزاری می باشد . ساختار سازمانی نمی تواند براحتی تغییر داده شود . نتایج عملی و سیاسی تغییرسازمانی در حیطه مسئولیت های مدیر پروژه نرم افزاری نمی باشد . اما تشکیلات مردمی که مستقیما در پروژه نرم افزاری دخیل هستند درحیطه کاری مدیر پروژه قراردارد.

بهترین ساختار تیمی به تعداد افراد تیم - سطح مهارتهای آنها و میزان پیچیدگی مساله بستگی دارد

انواع سازماندهی های ارائه شده

۱. غیر متمرکز دموکراتیک (DD)
۲. غیرمتمرکز کنترل شده (CD)
۳. متمرکز کنترل شده (CC)



انواع سازماندهی های ارائه شده جنبه ریاست (دموکراتیک+ کنترل شده) + مشورت گروهی برای حل مشکل (متمرکز+ غیرمتمرکز)

۱. سازمان غیر متمرکز دموکراتیک (DD) :

این تیم مهندسی نرم افزاری دارای یک رئیس دائمی نیست (موقتی). بلکه **هماهنگ کننده های** کار برای دوره کوتاهی در این سمت منصوب میشود و سپس افراد دیگری که ممکن است کارهای مختلفی را هماهنگ کنند جایگزین این هماهنگ کننده ها میشوند. و برای حل مسائل تصمیمات به صورت توافق گروهی است و ارتباط اعضا به صورت افقی است

۲. سازمان غیر متمرکز کنترل شده (CD) :

این تیم مهندسی نرم افزاری دارای یک **رئیس ثابت** میباشد که کارهای خاص و مدیران ثانوی که مسئولیت کارهای فرعی را برعهده دارند، هماهنگ و مشخص میکند اما **حل مشکلات به صورت یک کار گروهی** باقی می ماند اما **اجرای** راه حل های مشخص شده برای مشکل توسط رئیس تیم در میان گروه های مختلف تقسیم می شود. ارتباطات موجود میان گروه های فرعی و افراد به صورت هم سطح (افقی) می باشد. همچنین ارتباط عمودی در امتداد سلسله مراتب کنترل به وقوع می پیوندد .

۳. سازمان متمرکز کنترل شده (CC) :

حل مشکلات در سطوح بالا و ایجاد هماهنگی داخلی در میان تیم تحت نظارت و کنترل رئیس و اعضا تیم به صورت عمودی باشد.

هفت فاکتور مانتي که بايد در هنگام برنامه ريزی ساختار تیم های مهندسی نرم افزاری در نظر گرفته میشود:

- ۱- سختی مشکلاتی که باید مورد حل و فصل قرار گیرند. cc برای آسان و dd برای مشکل
 - ۲- اندازه برنامه حاصله در تعداد خطوط و جملات برنامه کامپیوتری یا امتیازات کارکردی (درجه ارتباطات) dd: درجه بالا:
 - ۳- مدت اشتغال تیم (مدت زمانی که اعضای تیم در کنارهم هستند) → در DD مهم است
 - ۴- تا چه اندازه میتوان مشکلات را به صورت پیمانه ای درآورد. → در DD هرچه کمتر بهتر
 - ۵- میزان اعتبار و کیفیت مورد نیاز که در ارتباط با سیستم باید ایجاد شود. بدون خط (پیمانه زیاد هرکس کار خودش): cc/cd
 - ۶- ثبات تاریخ تحویل (میزان اهمیت تاریخ تحویل)
 - ۷- میزان جامعه پذیری مورد نیاز برای پروژه (درجه ارتباط لازم با بیرون)
- از آنجایی که ساختار متمرکز کارها را سریعتر تکمیل میکند، بنابراین این ساختار در حل مشکلات ساده مفیدترین ساختار می باشد .
تیم های غیرمتمرکز نسبت به افراد ، راه حل های بهتر و بیشتری را بوجود می آورند . بنا براین احتمال موفقیت چنین تیم ها یی هنگام بررسی مسائل بسیار سخت بیشتر است .



مثال روزنامه

چهار الگوی سازمانی توسط Constantine جهت همکاری تیمهای مهندسی نرم افزار پیشنهاد گردیده است :

۱. **الگوی بسته :** این نمونه ، تیم مورد نظر را با توجه به سلسله مراتب سنتی قدرت و اختیارات تشکیل می دهد. چنین تیمهایی هنگام تولید نرم افزار که کاملاً مشابه با تلاش های قبلی است به خوبی عمل می کنند، اما احتمال خلاق بودن چنین تیم هایی هنگام کار کردن در یک نمونه بسته بسیار کم می باشد. (همه راه ها از بالا) (اگر پروژه تکراری باشد مناسب است)
۲. **الگوی تصادفی :** تیم را به صورت تصادفی و آزادانه تشکیل میدهد و به ابتکار فردی اعضاء تیم بستگی دارد. هنگامی که تحول ابداعات یا فناوری ها مورد نیاز باشد، تیمی که دارای نمونه تصادفی است به خوبی عمل خواهد کرد. اما چنین تیمی در صورتی که "عملکرد منظم و مرتب" مورد نیاز باشد با مشکل مواجه خواهد شد. (زمان تغییر تکنولوژی مناسب است)
۳. **الگوی باز :** این نمونه سعی دارد تیمی را به روشی تشکیل دهد که این تیم دارای بعضی از کنترل های مربوطه نمونه بسته باشد و در عین حال بیشتر ابداعات پدید آمده در صورت استفاده از نمونه تصادفی را نیز دارا باشد. کار در این نمونه به صورت دسته جمعی و همراه با ارتباط بیشتر و **توافق آرای مبتنی بر تصمیم گیری** و علائم تجاری تیم های نمونه باز انجام می شود. ساختارهای تیم نمونه باز برای یافتن راه حل برای مشکلات پیچیده بسیار مناسب هستند. اما برای حل این مشکلات به اندازه سایر تیم ها **سودمند نمی باشند**.
۴. **الگوی همگام :** این نمونه بر تشابه طبیعی مشکل (؟؟؟) و **تجزیه مساله** متکی می باشد و اعضای تیم را برای کار کردن بر روی بخش هایی از مشکل سازماندهی می کند و در طول این کار ارتباط فعال میان اعضاء تیم بسیار کم می باشد. روش دیگر مشهور "بی نظمی ابداعي" بود که کنستانتین ارائه داد که با اعمال تغییرات در دمکراتیک غیر متمرکز بود و کنترل هرج و مرج در آن مهم است

فاکتورهای لیستر که "موجب پیشرفت مسمومیت در محیط تیم می شوند." : مثال فوتبال

۱. یک محیط کاری آشفته (یا خیلی شاد) که در آن اعضاء تیم انرژی را اتلاف می کنند و بر اهداف لازم الاجرا کار توجه و تاکید ندارند.
۲. نا امیدی شدید که به وسیله پرسنل ، کار یا فناوری ایجاد می شود و این ناامیدی میان اعضاء تیم اختلاف ایجاد می کند.
۳. شیوه های تفکیک شده یا ناقص هماهنگی یا تعریف ناقص یا انتخاب نادرست ، مانعی برای تکمیل کار خواهد بود.
۴. تعریف نامشخص نقش کارکنان موجب از بین رفتن حس مسئولیت می شود.
۵. تعریف قرار گیری در معرض شکست مکرر و پیوسته که موجب از بین رفتن اعتماد به نفس و تنزل مسائل اخلاقی می شود.



راههای جلوگیری از انتخابهای نادرست در روند نرم افزاری

۱. مطمئن شدن از این امر که مشخصات نرم افزار مورد ساخت مطابق با سختی و عدم انعطاف روند انتخاب شده می باشد یا نه.
 ۲. اجازه دادن به تیم برای انتخاب روند با شناخت کامل از این مسئله که تیم ضمن انتخاب ، مسئول آن است که یک محصول با کیفیت را ارائه کند.
- هر یک از تیم های نرم افزاری با شکست های کوچکی مواجه می شوند . **کلید اصلی برای اجتناب از جو شکست** . نا کا می آن است که فنونی بر اساس تیم برای فید یک (بازگشت) و حل مشکل ایجاد شود. با این وجود شناخت تفاوت های انسانی اولین گام در راه ایجاد تیم همبسته می باشد.

هماهنگی و ارتباط مثال XML

قابلیت استفاده داخلی به صورت یکی از مشخصه های اصلی بسیاری از سیستم ها در آمده است. نرم افزار جدید باید با نرم افزار موجود ارتباط برقرار کند و با محدودیت های از پیش تعیین شده ای که از سویی سیستم یا محصول تحمیل می شود ، مطابقت داشته باشد.

این خصیصه های نرم افزاری مدرن ، یعنی مقیاس ، عدم اطمینان و قابلیت استفاده داخلی حقایق زندگی هستند. برای کنترل موثر این خصیصه ها ، تیم مهندسی نرم افزاری باید شیوه های مناسب و موثری را برای هماهنگ کردن افرادی که کار را با هم انجام می دهند ، ایجاد کند. برای تکمیل هماهنگی ، مکانیسم های لازم برای روابط رسمی و غیر رسمی میان اعضای تیم و روابط میان چند تیم مختلف باید مشخص شوند. روابط رسمی از طریق نوشتن نامه ، ملاقات رسمی و تقریباً از طریق سایر کانالهای ارتباطی غیر شخصی به وجود می آید. روابط غیر رسمی شخصی تر است.

روشهای ایجاد هماهنگی

شیوه های غیر رسمی و میان فردی

روابط الکترونیکی

شبکه میان فردی

رهافت رسمی و غیر رسمی

شیوه های رسمی و میان فردی



روشهای هماهنگی اعضای تیم

رہیافت های رسمی و غیر شخصی (رسمی؟؟؟؟) :

شامل اسناد و مدارک قابل تحویل مهندسی نرم افزار، نوشته های فنی ، زمان ارزیابی پروژه ، برنامه ها و ابزار کنترل پروژه ، بوجود آمدن تغییرات مجدد و مدارک مربوطه ، گزارشهای مربوط به پیگیری خطاها و اطلاعات سری.

شیوه های رسمی و میان فردی (اشخاص) :

بر فعالیت های تضمین کیفیت که در مورد محصول کاری مهندسی نرم افزار به کار برده شده است ، تأکید دارد. این شیوه ها شامل جلسات تجدید نظر رسمی ، بررسی مجدد طرح ها و کدها می باشد.

شیوه های غیر رسمی و میان فردی (اشخاص) :

این شیوه ها معمولاً شامل جلسات گروهی و تیم ها می باشد که به منظور پخش اطلاعات و حل مشکلات و جمع آوری اطلاعات در مورد نیازهای کارمندان و توسعه انجام شده ، برگزار می شود.

روابط الکترونیکی :

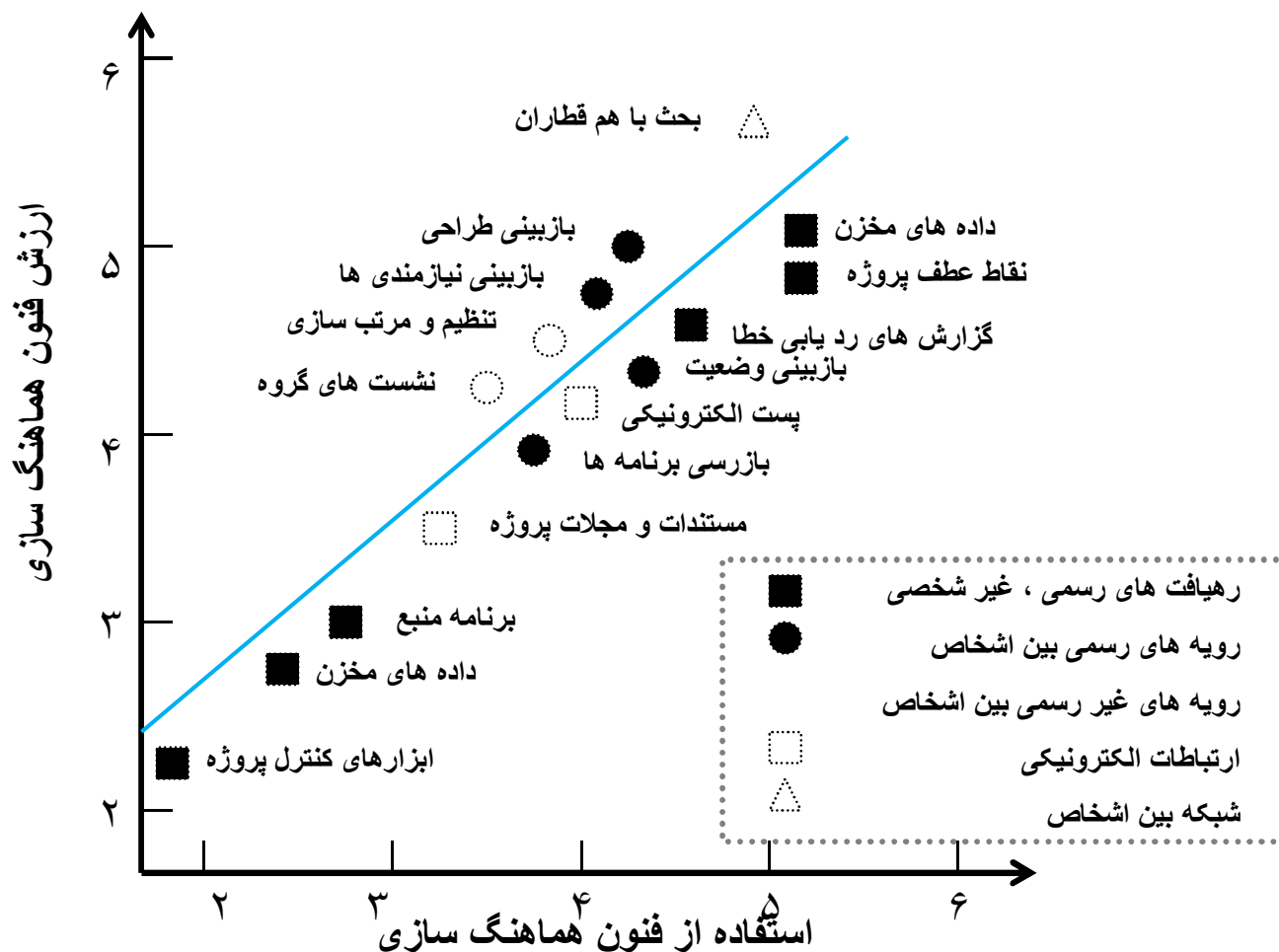
شامل پست الکترونیک ، برد های الکترونیکی پژوهشی و در حالت گسترده و پیشرفته شامل سیستم های گفتگو ویدئویی

شبکه میان فردی :

در اختیار قرار دادن تجارب قبلی اعضا

ارزیابی کارایی این تکنیک ها

برای ارزیابی کارایی این تکنیک ها در هماهنگ کردن پروژه **کارول و استریت** پروژه نرم افزاری را که شامل صدها نفر کارمند فنی مورد بررسی قرار داد. مشخص می شود که اندازه مشخص شده برای هماهنگ کننده ها و فنون مختلف روابطی در مقابل تعداد دفعات استفاده از آنها در طول یک پروژه طرح ریزی شده است. فنونی که در قسمت بالایی **رگرسیون** قرار گرفته اند ، با در نظر گرفتن **تعداد دفعات** آنها به عنوان فنون با ارزش و مفید در نظر گرفته می شوند. اما خطوطی که در پایین خط رگرسیون واقع شده اند ، از ارزش کمتری برخوردارند. جالب است بدانید شبکه ارتباطی بین فردی به عنوان تکنیکی که دارای بالاترین میزان هماهنگی و بالاترین اندازه ارتباط است ، مشخص شد. همچنین ذکر این نکته نیز حائز اهمیت است که مکانیسم های اولیه تضمین کیفیت نرم افزاری (بازبینی نیازمندی ها و طراحی) نسبت به ارزیابی های بعدی کد برنامه منبع (بازرسی های مربوط به کدها) از ارزش بیشتری برخوردار هستند.



محصول

باید محصول و مسئله ای را که قرار است حل نمایم در همان ابتدای شروع پروژه ، مورد بررسی قرار دهیم ، حداقل باید حوزه و حیطه و دامنه محصول تعیین گردد.

دامنه نرم افزار :

اولین فعالیت مدیریت پروژه تعیین دامنه نرم افزار است. که با پاسخ به سوالات زیر مشخص می شود:

سوالاتی برای تعیین دامنه نرم افزار:

۱. **بافت (جهت تعیین قیود لازم) :** چگونه می توان نرم افزاری ساخت که با سیستمی یا محصولی یا بافت تجاری بزرگتر هماهنگ باشد و قیود ناشی از این امر چه چیزهایی می باشد؟
 ۲. **اشیاء اطلاعاتی (ورودی/خروجی) :** کدام اشیاء داده ای است که مشتری به گونه ای واضح آنها را می بیند و درک می کند و به عنوان خروجی اصلی حاصل از نرم افزار بوجود می آورند ؟ برای ورودی چه اشیاء داده ای مورد نیاز است؟
 ۳. **کارکرد و عملکرد (عملیات) :** نرم افزار برای تبدیل داده های ورودی به داده های خروجی چه عملی را انجام می دهد؟ آیا ویژگی های عملکردی خاصی باید مورد بررسی قرار گیرد؟
- برای ساخت یک طرح پروژه قابل دفاع اولویت با شکستن مسئله برای حل آن می باشد.

تجزیه و شکستن مسئله :

تجزیه مشکل که در بعضی مواقع تفکیک سازی مشکل یا تشریح مشکل نامیده می شود، فعالیتی است که در مرکز تجزیه و تحلیل نیازمندی های نرم افزاری قرار دارد. در طول انجام فعالیت مربوط به مشخص کردن دامنه ، هیچ تلاشی برای تجزیه کامل مشکل انجام نمی شود. بلکه تجزیه در دو محدوده اصلی ذیل به کار برده میشود:

۱- زمینه های تجزیه : ۱- عملکردهایی که باید تحویل شوند. ۲- روند استفاده شده برای ارائه این تحویلها

با مشخص شدن صورت وضعیت دامنه ، اولین مرحله تفکیک سازی به صورت طبیعی اتفاق می افتد.



فرآیند

مراحل انجام پروژه: **a- انتخاب مدل b- ایجاد طرح اولیه و cpf ها طبق مدل c- تجزیه مسئله (توابع)**

مراحل کلی مشخص کننده فرآیند نرم افزار - **تعریف ، توسعه و پشتیبانی** - در مورد تمام نرم افزارها قابل اجرا خواهند بود. مشکل موجود در این مرحله انتخاب مدل فرآیند متناسب با نرم افزار می باشد که قرار است توسط تیم پروژه طرح ریزی شود.

مدل ترتیبی خطی	مدل ایجاد نمونه اولیه	مدل RAD	مدل افزایشی
مدل حلزوننی (Spiral)	مدل برنده - برنده (Win-Win)	مدل توسعه همزمان	
مدل توسعه مبتنی بر مولفه	مدل روشهای رسمی (Formal)	تکنیکهای نسل چهارم	

a مدیر پروژه باید تصمیم گیری کند که کدام یک از نمونه های فرآیند برای نیل به اهداف زیر مناسب می باشد:

- ۱- مشتریانی که این محصول را سفارش داده اند و مردمی که کار مربوط به آن را انجام خواهند داد.
- ۲- مشخصه های خود محصول
- ۳- محیط پروژه که در آن تیم نرم افزاری مشغول به کار است.

b پس از انتخاب مدل پردازشی ، تیم ، طرح اولیه پروژه را براساس مجموعه ای از فعالیت های مشترک و رایج چارچوب فرآیند تعریف خواهد کرد. پس از مشخص شدن طرح ابتدایی ، تفکیک و تجزیه فرآیند آغاز می شود. یعنی یک طرح کامل که منعکس کننده وظایف کاری مورد نیاز برای پر کردن چارچوب فعالیت ها است باید در این مرحله انجام شود.

c تجزیه همزمان با کارهای مدیریت شروع می شود و همانند برنامه ای لیست فعالیتها را نشان می دهد

cpf

تلفیق محصول و فرآیند :

برنامه ریزی پروژه با تعین محصول و فرآیند آغاز می شود ، و شامل یکسری فعالیت هایی تعریف شده سازمانی (زمینه کاری) است:

- **ارتباط با مشتری** : کارهایی که برای کشف موثر نیازمندیها میان توسعه دهنده و مشتری لازم می باشند.
- **طرح ریزی** : کارهایی که برای تعریف منابع ، خطوط زمانی و سایر اطلاعات مربوط به پروژه نیاز می باشد.
- **تجزیه و تحلیل خطرهای احتمالی** : کارهایی که برای ارزیابی خطرهای احتمالی فنی و مدیریتی ضروری می باشند.
- **مهندسی** : کارهایی که برای ساخت یک یا چند مورد از کاربرد های طرح لازم می باشند.
- **ساخت و واگذاری** : کارهایی که برای ساخت ، امتحان ، نصب و حمایت از کاربر لازم می باشند.
- **ارزیابی مشتری** : کارهایی برای دستیابی به بازخور مشتری بر اساس ارزشیابی انجام شده از کارهای نرم افزاری که در طول مرحله طراحی ایجاد و در طول مرحله نصب اجرا شده اند.

تلفیق محصول (مساله) و فرایند

اولین کار برای برنامه ریزی است که با انتخاب مدل فرآیند ، چارچوب روند معمول نیز با آن تطابق پیدا می کند. در هر موردی CPF فوق الذکر (در اینجا : ارتباط با مشتری ، برنامه ریزی ، تجزیه و تحلیل احتمال خطر ، طراحی ، ساخت ، انتشار ، ارزیابی مشتری) را می توان متناسب با نمونه حل کرد. این مسئله در مورد مدل های خطی ، مدل های افزایشی یا تکراری ، مدل های تکامل یافته و حتی در مورد مدل های سوارکردنی همزمان نیز صادق است. CPF ثابت و غیر معمول است و به عنوان پایه ای برای تمام کار های نرم افزاری که توسط سازمان نرم افزاری اجرا می شوند ، عمل می کند. برای هر سلول و هر تابع عمده بایست تاریخ شروع و پایان هر فعالیت ، محصولاتی که بایست تولید شوند و منابع مورد نیاز تعیین شوند

فعالیت های چارچوب مشترک فرآیند	ارتباط با مشتری	طرح ریزی	تحلیل ریسک	مهندسی
فعالیت های مهندسی نرم افزار				
کارکردهای محصول				
ورودی متن				
ویرایش و قالب بندی				
اصلاح خودکار				
آرایش صفحه و صفحه بندی				
شاخص بندی خودکار و TOC				
مدیریت فایل				
تولید مستندات				

پروژه

□ به منظور دستیابی به یک پروژه نرم افزاری موفق؟؟؟ و مدیریت يك پروژه، ابتدا باید به این مطلب پی برد که چه مشکلی وجود دارد و چگونه می توان آن مشکل را برطرف نمود.

□ **علائمی که از نظر جان ریل پروژه سیستم های اطلاعاتی در معرض خطا (۱۰ ریسک) قرار گرفته اند عبارتند از:**

- ۱. متخصصان نرم افزاری نیازهای مشتری خود را درک نمی کنند.
- ۲. حوزه عملکرد محصول ضعیف تعریف شده است.
- ۳. تغییرات به خوبی کنترل نمی شوند.
- ۴. فناوری انتخاب شده تغییر می یابد.
- ۵. نیاز های تجاری در حال تغییر هستند.
- ۶. موعد های مقرر و مهلت ها غیر واقعی هستند.
- ۷. کاربران مقاوم هستند.
- ۸. ضمانت این سیستم ها هرگز به صورت مناسب و درست حاصل نشده است.
- ۹. افراد متخصص با مهارت های مناسب در تیم پروژه وجود ندارد.
- ۱۰. مدیران و کارورزان از به کار بردن بهترین تمرین ها و درسهای فرا گرفته شده اجتناب می کنند.

□ **ریل رهیافت پنج قسمتی قضاوت صحیح را در مورد پروژه های نرم افزاری پیشنهاد می دهد :**

- ۱. کار را به روشی صحیح آغاز کنی.
- ۲. نیروی سوق دهنده پروژه را حفظ کنید. (داشتن نیروی محرک)
- ۳. پیشرفت حاصله را دنبال کنید.
- ۴. در پروژه خود تصمیم گیری صحیح و عاقلانه انجام دهید.
- ۵. یک تجزیه و تحلیل نیز پس از اتمام پروژه انجام دهید.

• مثال بسکتبال



اصل WWWWWHH یا W⁵HH پروژه منتهی

Boehm روشی را مطرح می نماید که در آن اهداف پروژه ، مسئولیت ها ، زمانبندی ها ، مدیریت و رویکردهای تکنیکی ، و منابع را با ذکر سؤالاتی مورد توجه قرار می دهد. این سؤالات عبارتند از :

چرا سیستم توسعه داده می شود؟

Why is the system being developed?

چه چیزی در چه موقع انجام شود؟

What will be done, by **When**?

چه شخصی مسئول چه کاری می باشد؟

Who is responsible for a function?

از لحاظ سازمانی در کجا قرار دارند؟

Where are they organizationally located?

از لحاظ تکنیکی و مدیریتی کار چگونه انجام می شود؟

How will the job be done technically and managerially?

از هر منبع چه میزان مورد نیاز است؟

How much of each resource is needed?

اصل WHH بوهم صرف نظر از پیچیدگی پروژه نرم افزاری قابل استفاده است. سؤالات فوق الذکر می تواند **طرحی عالی از برنامه ریزی را برای مدیران پروژه و تیم نرم افزاری فراهم کند**.

اقدامات بحرانی (حساس و لازم الاجرا)

شورای Air Lie لیستی از کارهای حساس مهم نرم افزاری را در ارتباط با مدیریت مبتنی بر عملکرد فراهم کرده اند. این کارها به صورت ثابت و پیوسته از سوی پروژه ها و سازمان های موفق نرم افزاری که عملکرد خط نهایی آنها دائما بیش از عملکرد متوسط صنایع می باشد ، مورد استفاده قرار گرفته و به عنوان کارهای حساس در نظر گرفته می شوند.

- **مدیریت رسمی ریسک و خطر:** برنامه ها برای ۱۰ خطر و ریسک عمده
 - **هزینه عملی و برآورد برنامه (زمانبندی):** برآورد هزینه تخمینی نرم افزار
 - **مدیریت پروژه مبتنی بر متریک ها (معیارها):** وجود معیارهای لازم برای تعیین مشکلات
 - **ردگیری مقادیر بدست آمده:** ارائه گزارشات ماهانه مقادیر مربوط به معیارها و...
 - **پیگیری معایب در مقابل اهداف کیفیتی:** ارائه گزارش پیرامون معایب
 - **آگاهی از برنامه ریزی افراد:** بررسی عملکرد کارکنان
- اگر تیم پروژه نرم افزاری نتواند به این سوالات پاسخ دهند یا پاسخ آنها به این سوالات به صورت ناقص باشد ، بررسی کامل و جامع کارهای پروژه ضروری خواهد بود.



فصل چهارم

متریک های پروژه و فرآیند نرم افزار

اندازه گیری اساس هر روش مهندسی از جمله مهندسی نرم افزار است (مثلا گرافیک/ادبیات/معماری...)

تعریف متریک

متریک های فرآیند و محصول نرم افزاری اندازه گیری های کمی هستند که به افرادی که با نرم افزار کار می کنند امکان می دهند تا نسبت به کارآمد بودن فرآیند و پروژه های نرم افزاری بینش پیدا کنند.

علت اهمیت متریک

اگر اندازه گیری صورت نپذیرد، قضاوت بر اساس ارزیابی ذهنی باشد

مراحل کار با متریک ها

با تعریف و جمع آوری مجموعه ای محدود از اندازه ها شروع شده و برحسب متریک اندازه گرا یا کارکرد گرا نرمال می شوند و نتایج مورد تحلیل قرار می گیرند

انواع متریکها :

- ۱- متریکهای پروژه محصولات (نرم افزار) ۲- متریکهای فرآیندها

دلایل اندازه گیری منابع / محصولات و فرآیندها

- | | |
|------------------------|---|
| ۱. مشخص کردن | برای دستیابی به فهم فرآیندها، محصولات، منابع و محیطها |
| ۲. ارزیابی کردن | تعیین وضعیت نسبت به برنامه ریزی و امکان فهم زمان انحراف پروژه ها و فرآیندها |
| ۳. پیش بینی کردن | برای انجام برنامه ریزی مورد نیاز است |
| ۴. پیشرفت کردن (اصلاح) | ایجاد پیشرفت با شناسایی موانع و نواقص و فرصتهای بهبود کیفیت |

واژه اندازه ها، متریکها و معیارها

لغت‌های measure (اندازه) و metric (متریک) غالباً به جای یکدیگر به کار می‌روند باید توجه داشت که با یکدیگر تفاوت دارند.

- **Measure اندازه** = یک شاخص کمی از اندازه، مقدار، ابعاد و ظرفیت یک صفت از یک محصول یا فرآیند را تعیین می‌کند.
- **metric (متریک)** = از ترکیب اندازه‌های منفرد، درکی از پروژه یا محصول حاصل می‌شود
- **Indicator (شاخص)** = یک معیار شاخص یک متریک یا ترکیبی از متریک‌هاست که به صورت نتیجه حاصل شده است

مثال:

تیم ۱ : ۵۰ نفر ساعت کار کرده است	به ازای هر نفر ساعت ۶ خطا (متریک)
تیم ۲ : ۱۰۰ نفر ساعت کار کرده است	به ازای هر نفر ساعت ۸ خطا (متریک)
نتیجه (شاخص): تیم ۱ بهتر بوده	

متریک‌ها در حوزه پروژه و فرآیند

در نتیجه جمع آوری مناسب متریک‌ها می‌توان :

۱. **تولید نشانگرهای فرآیندی** به یک سازمان مهندسی نرم افزار امکان می‌دهند تا ارزیابی کنند چه چیزی موثر و چه چیزی غیر موثر است. متریک فرآیند برای تولید نشانگر در سیرتاسیر پروژه‌ها و طی زمان‌های طولانی جمع آوری میشود. هدف آنها ایجاد علامتهای پروژه است که به بهبود طولانی مدت در فرآیند نرم افزاری منجر می‌شود.

۲. **تولید نشانگرها (علامتهای) پروژه در کوتاه مدت** به کار ما می‌آیند و به مدیر پروژه امکان می‌دهند:

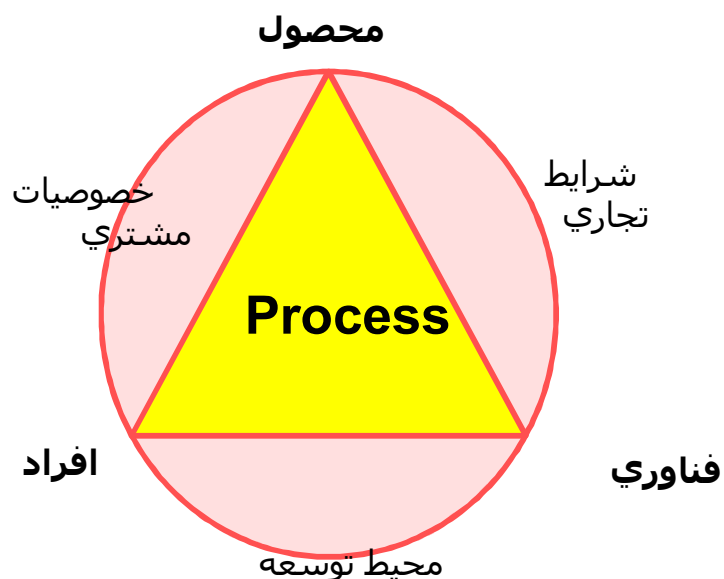
- ۱- خطرات بالقوه را پیدا می‌کند.
- ۲- زمینه های مشکل آفرین را چاره میکند.
- ۳- جریان کار یا وظایف را تنظیم می‌کند.
- ۴- وضعیت یک پروژه در حال اجرا را ارزیابی می‌کند.
- ۵- ارزیابی توانایی تیمهای پروژه برای کنترل کیفیت محصول

متریکهای فرآیند و اصلاح فرآیند نرم افزار

راههای منطقی برای پیشبرد هرگونه فرآیند

۱. اندازه گیری های خاص آن فرآیند
۲. توسعه ی مجموعه ای از متریکهای فرآیند
۳. استفاده از این متریک ها برای تهیه ی علامت ها و شاخص هایی که باعث ایجاد یک راهبرد برای پیشرفت شوند.

عوامل موثر بر کیفیت نرم افزاری و کارایی سازمانی



مهارت و انگیزه افراد بیشترین تاثیر
پیچیدگی محصول
فناوری یا روشهای مهندسی نرم افزار

مثلت
فرایند

محیط توسعه (ابزارها)
شرایط تجاری (مهلت ها)
خصوصیات مشتری (سهولت ارتباط)

دایره
شرایط محیطی

چگونگی اندازه گیری کارایی فرایند

به طور غیر مستقیم و توسط متریکهایی که از نتایج قابل حصول از فرآیند استخراج می شوند
برای داده های فرآیند استفاده های عمومی و خصوصی وجود دارد در حالت خصوصی یا شخصی متریکها ، نشانگری برای آن شخص هستند

انواع داده های قابل حصول از فرآیند

۱. متریکهای شخصی فرآیند و فلسفه فرآیند نرم افزار شخصی (PSP)

یک مجموعه ساخت یافته از توصیف های پردازشی / اندازه گیریها و روشهایی که می توانند به مهندسان در پیشبرد عملکرد شخصی شان کمک نمایند. این مجموعه به آنها نشان میدهد که چگونه فرآیند ها را تعریف کرده و کیفیت و بهره وری آن ها را اندازه گیری نمایند. یکی از اصول PSP آن است که **همه افراد با هم فرق می کنند** و روشی که برای یک مهندس مفید است شاید برای دیگری مناسب نباشد. دارای اهمیتند چراکه راهنمای مهمی برای اصلاح فعالیتهای هر مهندس نرم افزارند. مثل نرخ خطاهای فردی - خطاهای پیمانه ای یا خطاهای یافت شده ضمن توسعه

۲. متریکهای عمومی فرآیند

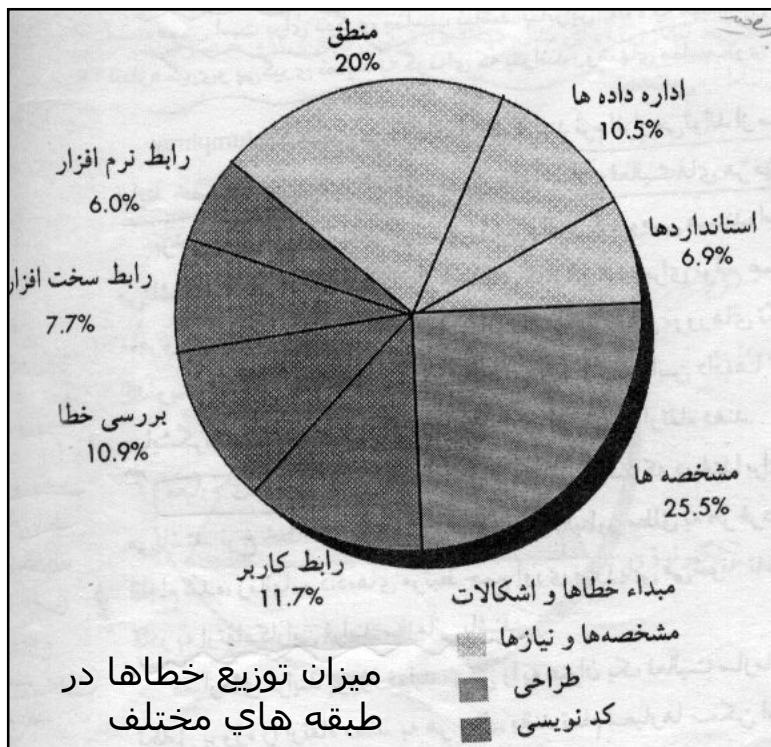
قادر به ارتقا کارایی فرایند سازمانی می باشند
و ممکن است برای افراد یا تیمی خصوصی باشند ولی برای همه اعضای تیم عمومی هستند
مثل نرخ خطاهای موجود در سطح پروژه

رهنمودهایی پیرامون جمع کردن متریکها نرم افزار

- استفاده از حساسیت سازمانی در زمان تفسیر متریکها
- امکان فیدبک و بازخورد برای افراد مسئول جمع آوری اندازه ها
- برای ارزیابی افراد ، متریک و شاخص استفاده نشود
- عدم استفاده از متریکها برای ترساندن و تهدید افراد بلکه برای دستیابی به اهداف و ارتقا فرآیند
- داده های نمایشگر مشکل را نباید با دید منفی نگریست بلکه در ارتقا به کار برد

پیشرفت آماری فرآیند نرم افزار (SSPI)

استخراج معیارها و نمایشگرها از متریکها برای ارتقا آماری فرایند نرم افزار لازم است.
در اصل SSPI استفاده از تحلیل خرابی و شکست نرم افزار برای جمع آوری تمام خطاهای مشخص شده در ضمن توسعه برنامه کاربردی ، سیستم یا محصول



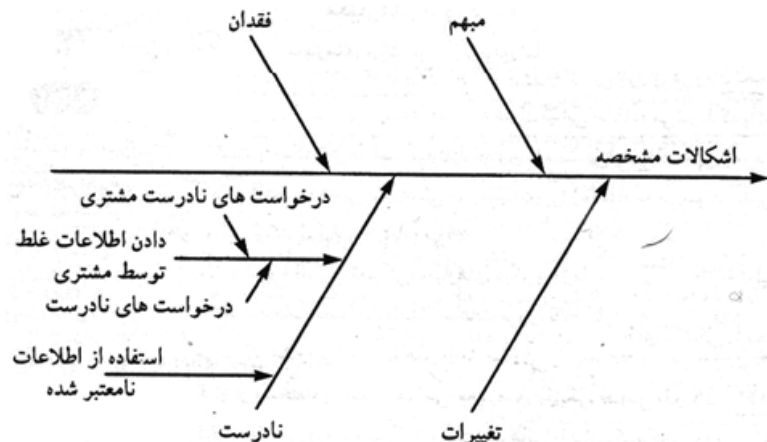
نحوه عملکرد تحلیل خرابی (شکست)

۱. همه خطاها و معایب بسته به منشاء شان **طبقه بندی** میشوند. (طبق مشخصاتشان)
۲. هزینه تصحیح هر خطا و عیب ثبت می شود.
۳. **تعداد خطاها و معایب در هر طبقه** شمرده میشود و آنها به ترتیب از بالا به پایین رده بندی می شوند.
۴. **هزینه کلی خطاها و معایب در هر طبقه** محاسبه میشود
۵. تحلیل داده ها برای یافتن **طبقه ای که بیشترین هزینه را برای سازمان دارد**
۶. **ایجاد طرحهایی برای حذف خطاها** از آن طبقه

نموار استخوان ماهی

برای کمک به تشخیص داده در هر طبقه از نمودار درصد عیوب نمودار استخوان ماهی پیشنهاد شده است

خط مرکزی : فاکتور کیفی مورد نظر
خطوط مورب: دلایل ایجاد مشکل





متریکهای پروژه

دانشتیم متریکها **فرآیند** برای مقاصد **راهبردی** (و بلند مدت) استفاده می شدند
اندازه های پروژه **تاکتیکی** هستند، یعنی متریکهای حاصله برای تطبیق جریان کار پروژه و فعالیتهای تکنیکی توسط یک مدیر پروژه و تیم نرم افزار مورد استفاده قرار می گیرند
در **زمان تخمین**، برای اولین بار از متریکهای پروژه استفاده می شود، چرا که متریکهای بدست آمده در پروژه های قبلی مورد استفاده قرار می گیرند.
با شروع پروژه و فعالیتهای تکنیکی، متریکهای دیگر پروژه مورد اهمیت قرار می گیرند، و اندازه هایی چون تعداد صفحات اسناد، تعداد خطوط کد و... محاسبه می شوند

اهداف متریک های پروژه: (جنبه های متریکهای پروژه)

۱. متریک ها برای به حداقل رساندن زمان برنامه پیشرفت از طریق انجام تطبیق های لازم برای پرهیز از تاخیرها و کم کردن خطرات و مشکلات بالقوه بکار می روند.
۲. متریکهای پروژه به منظور سنجش کیفیت محصول بصورت مستمر و اصلاح رهیافت فنی برای بهبود کیفیت در صورت لزوم مورد استفاده قرار می گیرد.

مواردی که یک پروژه بایست اندازه گیری کند

۱. **ورودی ها**-اندازه منابع (یعنی افراد و محیط) لازم برای انجام کار : ۱۰ نفر برنامه نویس ، ۳ تا کامپیوتر
۲. **خروجی ها**-اندازه گیری های محصولات کاری یا قابل حمل که طی فرآیند مهندسی نرم افزار تولید میشوند. ۵: تا فرم، ۲ تاجدول
۳. **نتایج**-اندازه هایی که موثر بودن کالاهای قابل حمل و تحویل را بیان میکنند. : ۸ خطا در قسمت نمایش و...

اندازه گیری نرم افزار

۱. اندازه گیری مستقیم: شامل هزینه و تلاش به کار رفته می باشد. مانند: خطوط کد تولیدی - سرعت اجرا - اندازه حافظه
۲. اندازه گیری غیر مستقیم عبارتند از: عملکرد - کیفیت - پیچیدگی - کارایی - قابلیت اطمینان و نگهداری

دلیل نیاز به نرمال سازی

عوامل زیادی بر تولید نرم افزار تاثیر می گذارند و نمی توان به صورت مستقیم متریکها را باهم مقایسه کرد
مثلا تیم A در يك پروژه ۱۸۰ خطا دارد و تیم B در يك پروژه ۲۲۰ خطا دارد کداميك بهترند؟ نمی توان جواب داد چون به میزان پیچیدگی پروژه و ... نیز وابسته است

اگر این داده ها نرمال شوند متریکهایی برای مقایسه حاصل می شود

انواع متریکهای نرمال شده ۱- مبتنی بر اندازه ۲- مبتنی بر کارکرد

Project	LOC	میزان فعالیت	مستندات (\$000)		Errors	عیب	افراد
alpha	12,100	24	168	365	134	29	3
beta	27,200	62	440	1224	321	86	5
gamma	20,200	43	314	1050	256	64	6
•	•	•	•	•	•		
•	•	•	•	•	•		
•	•	•	•	•	•		

۱- متریکهای مبتنی بر اندازه (اندازه گرا)

این متریکها از طریق نرمال کردن متریکهای کیفیت و با در نظر گرفتن اندازه و حجم نرم افزار تولید شده بدست می آیند
برای هر سازمان نرم افزاری (پروژه) می توان جدولی از اندازه های مبتنی بر اندازه گیری ارائه داد
این اطلاعات شامل تمام فازهای ساخت پروژه هستند
عیبها بعد از تحویل محصول مشخص می شوند و خطاها در زمان ساخت
از تعداد خطوط برنامه (LOC) می توان برای نرمال سازی داده های اولیه استفاده کرد

متریکهای نرمال شده بر اساس تعداد خطوط برنامه

۱. خطاها در kloc (هزار خط برنامه)

۲. تعداد عیب ها در هر هزار خط

۳. هزینه هر خط برنامه LOC

۴. تعداد صفحات مستندات به ازای هر KLOC

متریکهای نرمال شده بر اساس میزان فعالیت

۵. تعداد خطا ها به ازای هر نفر-ماه

۶. تعداد خطوط برنامه به ازای هر نفر-ماه

۷. هزینه هر صفحه از مستندات

ایده موافقین با استفاده از متریکهای چون LOC شمارش آن برای اغلب پروژه ها ساده و روشن است

ایده مخالفین با استفاده از متریکهای چون LOC } تعداد خطوط به زبان برنامه نویسی وابسته است و ممکن است طراحی خوب
تعداد خطوط برنامه را بکاهد. در زبانهای غیر رویه ای قابل استفاده نیستند

۲- متریکهای مبتنی بر کارکرد (کارکرد گرا)

از یک معیار کار کردی که بوسیله برنامه نرم افزاری بعنوان یک ارزش نرمال سازی تحویل می گردد ، استفاده می نمایند .
به علت عدم امکان اندازه گیری مستقیم عملکرد از طریق اندازه های مستقیم ، اندازه گیری غیر مستقیم برای عملکرد
محاسبه می کنیم

با استفاده از یک رابطه تجربی بر اساس اندازه های قابل شمارش (مستقیم) دامنه اطلاعات نرم افزاری و ارزیابی
پیچیدگی نرم افزاری بدست می آیند.

$$FP = \text{count total} \times [0.65 + 0.01 \times \sum(F_i)]$$

امتیازات عملکردی (Fp-function point) با تکمیل جدول محاسبه می شوند. مقادیر جدول عبارتند از:

۱- **تعداد ورودی کاربر:** هر ورودی که داده های محوری و مورد نیاز برنامه را فراهم آورد. داده های ورودی باید با سوالاتی که جداگانه
پرسیده می شوند فرق داشته باشند .

۲- **تعداد خروجی کاربر:** هر خروجی (به گزارشها ، بررسی ها ، پیام های خطا و ... اشاره دارد) که اطلاعات مبتنی بر برنامه کاربردی
نرم افزار را فراهم آورد .

۳- تعداد در خواستهای کاربر : هر ورودی مرتبط که باعث بوجود آوردن چندین پاسخ نرم افزاری فوری به شکل خروجی مرتبط می شود.

۴- تعداد فایلها : هر نوع فایل اصلی منطقی (گروه بندی منطقی داده هایی که ممکن است یکی از بحثهای یک پایگاه داده های بزرگ یا یک فایل جدا باشند .)

۵- تعداد رابطه های خارجی : هر رابطه قابل خوانده شدن توسط ماشین (فایلهای داده ها بر روی نوار یا دیسک) که برای انتقال اطلاعات به یک سیستم دیگر بکار می روند .

ضرایب وزنی

پارامتر مدیریت	تعداد	ساده	متوسط	پیچیده	
تعداد ورودی های کاربر	<input type="text"/>	3	4	6	= <input type="text"/>
تعداد خروجی های کاربر	<input type="text"/>	4	5	7	= <input type="text"/>
تعداد درخواست های کاربر	<input type="text"/>	3	4	6	= <input type="text"/>
تعداد فایلها	<input type="text"/>	7	10	15	= <input type="text"/>
تعداد رابطه های خارجی	<input type="text"/>	5	7	10	= <input type="text"/>
Count total					→ <input type="text"/>

تعین پیچیدگی

وقتی داده ها جمع آوری شدند ، یک مقدار پیچیدگی با هر شمارش همراه می شود ، که تعیین آن تا حدی ذهنی است . و شامل ۱۴ سوال در رابطه با پروژه ، با امتیازات مابین ۰ (بی اهمیت) تا ۵ (خیلی مهم)

۱- آیا سیستم به پشتیبان و احیا و بازیابی فایل اطمینان نیاز دارد؟

۲- آیا ارتباطات داده ها مورد نیاز است ؟

۳- آیا عملگردهای پردازشی توزیع شده وجود دارند ؟

۴- آیا عملکرد آنها ضروری و بحرانی است ؟

۶- نیاز به تراکنش برای ورود داده های آنلاین؟

۷- آیا فایلهای اصلی بطور متصل و روی خط به هنگام سازی می شوند ؟

۸- آیا فرآیندهای درونی پیچیده است ؟ ۹- آیا تبدیلهای و نصب چندگانه در سازمانهای مختلف طراحی شده است ؟

۱۰- امکان استفاده مجدد از کد وجود دارد؟ ۱۱- آیا سیستم برای نصب چندگانه در سازمانهای مختلف طراحی شده است ؟

۱۲- امکان سهولت تغییرات ؟ ۱۳- پیچیدگی ورودی/خروجی/فایلها و درخواستها؟

۱۴- آیا سیستم به ورود داده ها روی خط(آنلاین) نیاز دارد؟

بعد از محاسبه امتیازات عملکردی برای نرمال کردن اندازه ها ، جهت کیفیت و بهره وری نرم افزار و دیگر ویژگیها بر حسب FP (امتیاز کارکردی) موارد زیر نیز استفاده می شود

۱. تعداد خطاها به ازای هر امتیاز کارکردی
۲. تعداد عیوب و نواقص به ازای هر امتیاز کارکردی
۳. تعداد صفحات مستندات به ازای هر امتیاز کارکردی
۴. تعداد امتیازات کارکردی به ازای هر نفر ماه

متریکهای توسعه یافته امتیاز کارکردی (امتیاز ویژگی / امتیاز کارکردی سه بعدی)

استفاده از امتیاز کارکردی که فاقد بعد عملیاتی و رفتاری است و بر بعد داده تاکید دارد در سیستمهای مهندسی و جاسازی شده چندان مناسب نیست

- ۱ يك امتیاز کارکردی توسعه یافته به نام **امتیاز ویژگی** به وجود آمد
امتیاز ویژگی ، کاربردهای طبقه بندی شده ای را اندازه گیری می کند که در آنها پیچیدگی الگوریتم زیاد است (مثل برنامه های بلادرنک، کنترل فرایند و جاسازی شده)
برای محاسبه امتیاز ویژگی ، مقدار امتیاز کارکردی با ضریبی محاسبه می شود به علاوه امتیاز ویژگی ، مشخصه جدیدی از نرم افزار به نام **الگوریتم** را نیز **شمارش** می کند، يك الگوریتم به صورت مسئله ای محدود شده محاسباتی مشتمل بر يك برنامه خاص تعریف می شود. مثل معکوس کردن ماتریس و ...
- ۲ توسعه دیگری برای امتیاز کارکردی توسط شرکت بونینگ ارائه شده ، **امتیاز کارکردی سه بعدی** نام دارد
در این روش علاوه بر بعد داده ، **بعدهای کنترلی و عملیاتی** را مشتمل می شود

ویژگیهای این روش عبارتند از : شمارش شده (قابل شمارش) ، اندازه گیری شده (کمی بودن) ، تبدیل شده (قابل تبدیل) به اندازه ای که معیاری برای قابلیت عملکرد تحویل شده توسط نرم افزار تهیه می کند .

بعد داده ها : در (مثل متریکهای مبتنی بر کارکرد)، تعداد داده های باقیمانده (فایلها) ۶ و داده های بیرونی همراه با اندازه های پیچیدگی بکار می روند .

بعد عملیاتی : از طریق بررسی تعداد عملیاتهای داخلی که برای تبدیل ورودی به داده های خروجی مورد نیاز هستند.

بعد کنترل : از طریق شمردن تعداد جابجایی های بین وضعیتها اندازه گیری می شود .



وضعیت در بعد کنترلی :

نوعی حالت رفتاری بیرونی حالت مشاهده را نشان می دهد .

انتقال در بعد کنترلی :

در نتیجه حادثه ای که سبب می شود نرم افزار یا سیستم حالت رفتاری اش را تغییر دهد ، بوجود می آید .

مثال

یک تلفن بی سیم دارای نرم افزاری که از عملکردهای شماره گیری خودکار پشتیبانی می کند :
فشاردن دکمه شماره گیری خودکار ، نمایشگر یک کد را نشان می دهد که بر مخاطب دلالت می کند . نرم افزار بلافاصله انرا به وضعیت شماره گیری تبدیل می کند . یعنی تغییر حالت صورت می پذیرد
در هنگام محاسبه امتیازات عملکردی 3D به انتقالها یک مقدار پیچیدگی نمی دهیم و از رابطه زیر برای محاسبه آن استفاده می کنیم :

$$\text{Index} = I + O + Q + F + E + T + R$$

که همگی پارامترها مقادیر وزن دار پیچیدگی را برای عناصر فوق الذکر نشان می دهند :
ورودی ها ، خروجی ها ، درخواست ها ، ساختار داده های درونی ، فایل های بیرونی ، تبدیل ، جابجایی ،
که هر مقدار وزن دار پیچیدگی با استفاده از رابطه زیر محاسبه می شود :

Low | Average | High

$$\text{مقدار پیچیدگی} = \text{Nil Wil} + \text{Nia Wia} + \text{Nih Wih}$$

که N ها **تعداد** وقوع عنصر i را برای هر سطح پیچیدگی (کم/متوسط/زیاد) نشان می دهند
و W ها **وزن** های معادل آن هستند .

ایده موافقین روشهای مبتنی بر کارکرد (fp): به زبان برنامه نویسی وابسته نیست و برای زبانهای غیر رویه ای مناسب است

ایده مخالفین روشهای مبتنی بر کارکرد (fp): نیازمند تردستی هستیم و عینی نیست و گاهی ذهنی است.



تطبیق دو روش متریک (رابطه بین کارکردگرا و اندازه گرا)

ارتباط بین خطوط برنامه و امتیازات عملکردی وابسته به زبان برنامه نویسی استفاده شده برای پیاده سازی نرم افزار و کیفیت طراحی است.
میزان عملکردی را که به وسیله برنامه کاربردی حاصل شود، می توان با جزء به جزء نوشتن قطعات اصلی، داده های ورودی یا خروجی تخمین زد
معیارهای LOC و FP برای بدست آوردن معیارهای قابلیت تولید استفاده می شوند.

زبان برنامه سازی	میانگین تعداد خطوط به ازای هر امتیاز کارکردی LOC / FP
Assembly Language	320
C	128
COBOL	106
FORTRAN	106
Pascal	90
C++	64
Ada 95	53
Visual Basic	32
Smaltalk	22
Powerbuilder (code generator)	16
SQL	12

• تعداد خطوط C++ تقریباً
۱.۶ برابر خطوط FORTRAN
، امتیاز و قابلیت عملکردی
را ایجاد می کند .

متریکهای کیفیت نرم افزار

کیفیت برنامه کاربردی فقط با موارد زیر بدست می آید

- مفید بودن نیازمندیهای توصیف کننده مشکل
- مفید بودن کد برنامه قابل اجرا

مهندس نرم افزار خوب از اندازه گیری برای موارد زیر استفاده می کند

- ✓ ارزیابی کیفیت تحلیل
 - ✓ کد منبع
- مدلهای طراحی
آزمونهایی که در موقع ساخت نرم افزار بوجود آمده اند

اولین قدم در سطح پروژه برای ارزیابی کیفیت ، اندازه گیری خطاها و اشکالات است

متریکهای زیر در **خصوص تاثیر و سودمندی فعالیتهایی** که متریکهایی برایشان به کار رفته ، بصیرت و آگاهی ایجاد می کنند:

➤ تعداد خطاهای محصول کاری به ازای هر امتیاز کارکردی (نیازمندیها یا طراحی)

➤ خطاهای کشف شده به ازای هر ساعت بازبینی

➤ تعداد خطاهای کشف شده به ازای هر ساعت آزمون

نگاهی اجمالی بر فاکتورهای موثر بر کیفیت

جنبه های بررسی کیفیت در نرم افزار

۱. راه اندازی محصول

۲. بررسی محصول (تغییر در آن)

۳. انتقال محصول (اصلاح برای محیط دیگر)

بررسی رابطه بین عوامل کیفیتی و جنبه های دیگر مهندسی نرم افزار :

(۱) ارائه چارچوب مکانیسمی برای اولویت بندی کیفیتها توسط مدیر پروژه

(۲) ارائه ابزاری برای ارزیابی کمیتی از نحوه توسعه و پیشرفت نسبت به اهداف کلی

(۳) ارائه یک چارچوب ارتباط بیشتر یا پرسنل QA در تلاش برای پیشرفت

(۴) استفاده از معیارهای کیفیت پایسن (شاخص) برای شناسایی استانداردهایی که بایست تقویت شوند



اندازه گیری کیفیت

اندازه های فراوانی از کیفیت نرم افزار وجود دارد ولی قابلیت زیر معیار های مفیدی ابعاد مختلف اندازه گیری هستند:

- ❖ **تصحیح (صحت)**
- ❖ **پیوستگی (جامعیت)**
- ❖ **تعمیر و نگهداری**
- ❖ **قابلیت استفاده**

□ **صحت** : برنامه باید به درستی کار کند ، صحت تعیین درجه ای است که نرم افزار کار خود را انجام می دهد رایجترین اندازه برای قابلیت اصلاح : تعداد نقصها (عدم انطباق با نیازمندیها) در هر هزار خط برنامه .

□ **قابلیت نگهداری** : تلاش بیشتری می طلبد که همان سهولت تصحیح برنامه در هنگام مواجه شدن با یک خطا، انطباق در صورت تغییر یا ارتقا در صورت نیاز کاربر است

که متوسط زمان تغییر (MTTC) یک متریک ساده مبتنی بر زمان برای اندازه گیری آن می باشد. قابلیت نگهداری اگر بالا باشد می طلبد که زمان تغییر پایین باشد

□ **جامعیت (تمامیت)** : توانایی یک نرم افزار را برای مقاومت در برابر دستبردها به برنامه ها ، داده ها و اسناد ، می سنجد . برای اندازه گیری جامعیت دو ویژگی باید بررسی شود :

- تهدید : بروز یک حمله از نوعی خاص و در محدوده زمانی مشخص
- امنیت : احتمال دفع نوعی خاص از حمله

$$[(\text{امنیت} - ۱) * (\text{تهدید} - ۱)] = \text{تمامیت}$$

□ **قابلیت استفاده (سهولت کاربرد)** : اگر برنامه قابلیت استفاده و سازگاری با مشتری نباشد ، دچار نقص خواهد شد ، حتی در صورتیکه عملکردهای آن ارزشمند باشند . و برحسب ۴ ویژگی اندازه گیری می شود:

۱. مهارت فیزیکی یا هوشی که برای یادگیری سیستم مورد نیاز است .
۲. زمان مورد نیاز برای ماهر شدن در استفاده از سیستم .
۳. افزایش خالص بهره وری که موقعی اندازه گیری می شود که سیستم توسط فردی که بطور متوسط کارایی دارد مورد استفاده قرار گیرد .
۴. یک ارزیابی ذهنی و معقول از **دیدگاههای کاربران نسبت به سیستم** .



کارایی رفع نقص DRE

يك متریک کیفیت که برای سطح پروژه و فرآیند مفید است. یک اندازه از توانایی جدا کردن فعالیتهای کنترل و تضمین کیفیت در موقعی بشمار می آید که در همه فعالیتهای چار چوب فرآیند به کار رفته باشد .

به طور ایده آل مساوی ۱ است . $DRE = E / (E + D)$

که E تعداد خطاهایی است که قبل از تحویل نرم افزار به کاربر نهایی مشاهده شده

و D خطاهایی است که بعد از تحویل یافت شده اند و در حالت عادی بزرگتر از ۰ است . - مقدار ایده آل DRE ، ۱ است

DRE را می توان راهی برای ارزیابی توانایی تیم در پیدا کردن خطاها قبل از انتقال به مهندسی نرم افزار یا فعالیت بعدی دانست .

اگر DRE بتواند ساختاری رابه وجود آورد که خطاها را طی فعالیتهای پیدا کرد و رفع نمود به صورت $DRE = E_i / (E_i + E_{i+1})$ بیان می شود که E_i تعداد خطاهای یافت شده در مرحله i ام و E_{i+1} تعداد خطاهای یافت شده ناشی از مرحله i در مرحله i+1 است

متریک های انسجام و جامعیت در فرایند مهندسی نرم افزار

اندازه گیری هنوز در شرکتها باب نشده است در این بخش چندین استدلال برای متریک ها را بررسی کرده و روشی را برای جایگزین کردن یک برنامه ی جمع اوری متریک ها در یک سازمان مهندسی نرم افزار ارائه می شود.

استدلالی بر متریک های نرم افزاری

علت به وجود آمدن اندازه گیری؟

اگر اندازه گیری انجام نشود راه دیگری برای ارزیابی پیشرفت وجود ندارد و بدون پیشرفت شکست حتمی است

وظیفه اندازه گیری؟

فهمیدن وضعیت فعلی و يك مبنای پردازشی برای تعیین اهداف بیشتر

سوالاتی که در سطح پروژه و تاکتیکی وجود دارد و اندازه گیری سبب پاس شدن آنها می شود

کدامیک از تجهیزات کاربران تغییر می کند؟ برای هر پیمانانه چه تعداد آزمون خطا لازم است؟

احتمال خطا در کدام پیمانانه ها بالاست؟

هنگام آزمون انتظار چند خطا وجود دارد؟

استقرار خط مبنا (ایجاد یک مبنا)

از طریق تعیین یک مبناي متریک، می توان به فواید بسیاری در سطح فرایند، پروژه و محصول دست یافت
مبناي متریک های حاوی اطلاعات جمع آوری شده از پروژه های قبلی توسعه نرم افزار است و می تواند ساده و یامثل
یک مبناي جامع که حاوی دهها اندازه گیری پروژه و متریک های مشتق از آنهاست، باشد.
ویژگیهای خط مبنا :

۱- داده ها صحیح باشند و حدسي نباشند

۲- داده را می توان از هر تعدادي پروژه جمع آوري کرد

۳- اندازه ها بایست با هم تطابق داشته باشند (عدم شمارش کد HTML) ۴- برنامه ها بایست مشابه باشند

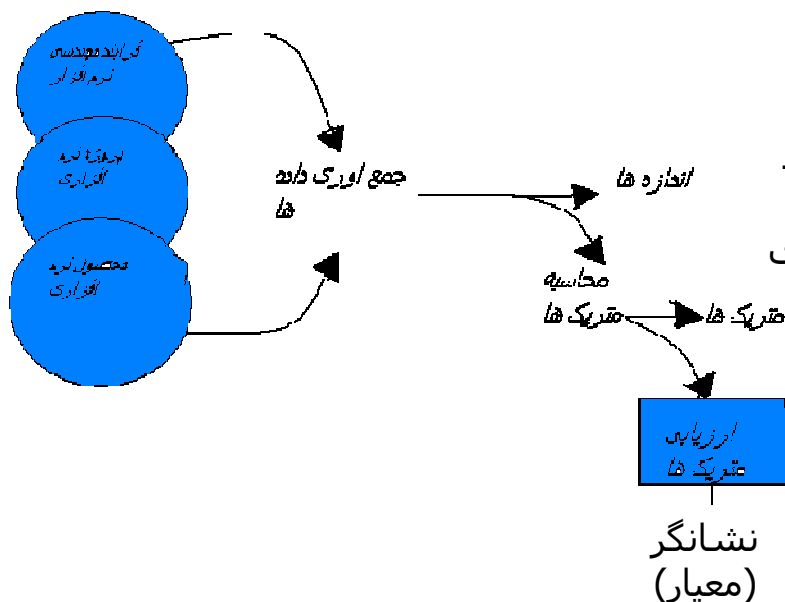
جمع آوری ، محاسبه و ارزیابی متریک ها

• در حالت ایده ال ، داده هایی که برای تعیین یک خط مبنا مورد نیاز هستند، بطور مستمر جمع آوری می شوند

• زمانی که اندازه ها جمع آوری می شوند، استفاده از متریک ها میسر میشود

• متریک ها می توانند دامنه وسیعی از متریک های fp یا loc و دیگر متریک های کیفیت و مبتنی بر پروژه را گسترش دهند. در نهایت اینکه متریک ها باید طی مراحل برآورد، کار تکنیکی، کنترل پروژه و بهبود فرایند، مورد ارزیابی قرار گیرند. ارزیابی متریک، دلایل پنهان در نتایج حاصله را آشکار می کند و مجموعه شاخص هایی را برای راهنمایی پروژه یا فرایند بوجود می آورد.

• نمودار جمع آوري متریک ها براي ایجاد خط مبنا



توسعه و تکمیل متریکها و GQM (هدف- سوال - متریک)

یک سازمان برای دارا بودن یک برنامه اندازه گیری صحیح که بیهوده نباشد باید دارای اجزا زیر باشد:

۱. فرایندی برای تولید اهداف (از طریق بیان چند سوال)
۲. فرایندی برای تبدیل اهداف به داده های پروژه ای.
۳. فرایندی برای تفسیر داده های پروژه ای

• GQM چیست؟

یکی از تلاشهای اولیه برای توسعه مجموعه ای از اندازه گیریها می باشد که برای نرم افزار قابل استفاده است و به الگوی پیشرفت نرم افزاری مربوط میشود.

• مراحل الگوی پیشرفت نرم افزار

۱. تخمین در مورد یک پروژه و محیط آن، تعین اهداف کیفیتی برای پروژه و انتخاب ابزارهای مناسب، روشهای مدیریتی و فناوری هایی برای ان اهداف تا شانس برآورده شدن را داشته باشد.
۲. فرایند اجرای یک پروژه و نظارت بر داده های مربوط به این اهداف کیفیتی این کار در رابطه با فرایند عملکرد بر اساس داده ها در موقع برآورده نشدن اهداف کیفیتی توسط مدیر پروژه انجام می شود.
۳. فرایندی برای تحلیل داده های جمع اوری شده در مرحله دوم بمنظور ارائه پیشنهاد برای پیشرفت بیشتر این فرایند مستلزم جستجو بدنبال مشکلات در جمع اوری داده ها و مشکلات در بکارگیری راهنمایی های کیفیتی ومشکل در تفسیر داده ها میباشد.

• ابعاد هدف (GOAL) در GQM؟

در سه مجموعه بازسازی می شود که عبارتند از : هدف ، جنبه و محیط

مدیریت تغییرات: کنترل آماری فرآیند

با توجه به اینکه پارامترهای زیادی روی فرایند تولید نرم افزار تاثیر دارند ، لذا متریکهای بدست آمده از يك پروژه به پروژه دیگر تغیر می کنند.

نمودار کنترل:

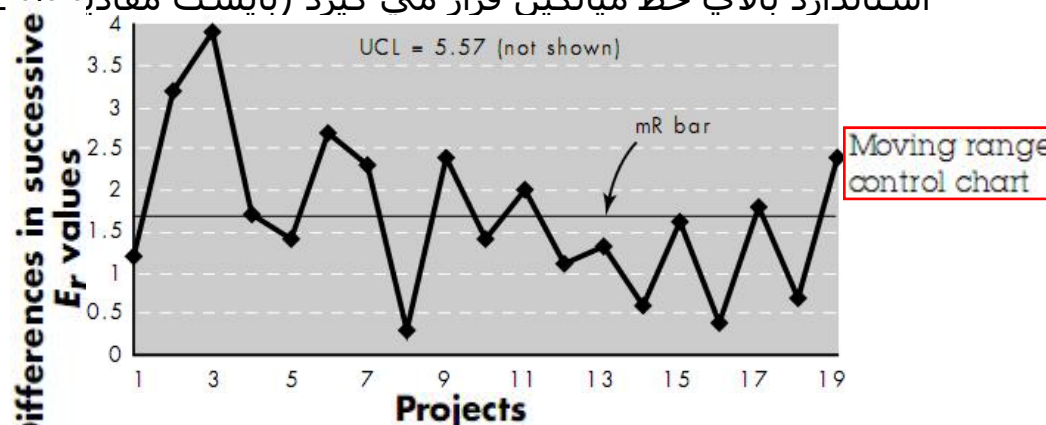
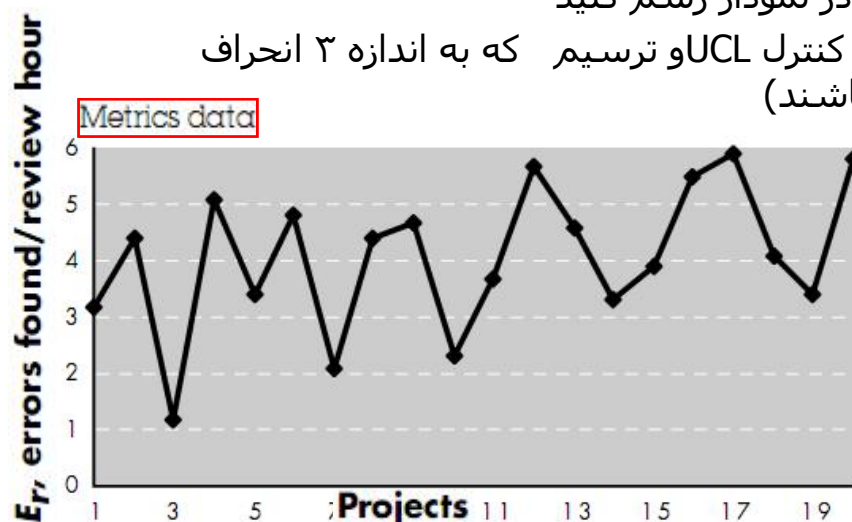
يك روش گرافیکی برای تعیین اینکه آیا تغییرات در داده های متریکها، معنی دار هستند؟
این تکنیک افراد علاقه مند به بهبود فرایند نرم افزاری را قادر می سازد که تعیین کنند که آیا پراکندگی (تغییرات) و مکان (میانگین حرکت) متریک های فرایند پایدار (فقط شامل تغییرات طبیعی و کنترل شده) می باشند یا غیر پایدار (تغییر غیرقابل کنترل و معیارها برای پیشگویی کارایی نمی تواند استفاده شوند)

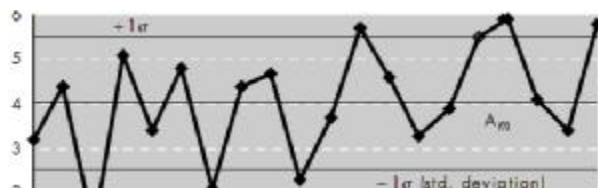
انواع نمودار کنترلی:

۱. نمودار کنترل طیف حرکت (MR)، ۲. نمودار کنترل فردی (مستقل) (NPL)

روش رسم نمودار کنترلی طیف حرکت:

- محاسبه رنج تغییرات: قدر مطلق تفاوتی متوالی بین هر دو زوج نقاط داده ، و رسم این رنج های متغیر روی نمودار
- محاسبه میانگین رنج تغییرات، این میله MR را به عنوان خط مرکزی در نمودار رسم کنید
- میانگین حاصله را در ۳.۲۶۸ ضرب کرده، و آنرا به عنوان خط حد بالای کنترل UCL و ترسیم که به اندازه ۳ انحراف استاندارد بالای خط میانگین قرار می گیرد (بایست مقادیر UCL باشند)





روش رسم نمودار کنترلی فردی (افراد):

- (a) مقادیر متریک های فردی را رسم می کنیم
- (b) برای مقادیر متریک ها مقدار متوسط (A_m) را محاسبه می کنیم (میانگین متریکها)
- (c) مقدار میانگین (MR) برای مقادیر طیف حرکتی را در عدد ۲.۶۶۰ ضرب می کنیم و مقدار متوسط (A_m) بدست آمده در مرحله قبل را با آن جمع می کنیم. این نتایج حد بالای فرایند (UNPL) خواهند بود. آن را رسم می کنیم
- (d) مقدار میانگین (MR) برای مقادیر طیف حرکتی را در عدد ۲.۶۶۰ ضرب می کنیم و مقدار متوسط (A_m) را از آن کم می کنیم. این نتایج حد پایین فرایند طبیعی (LNPL) خواهند بود. آن را رسم می کنیم
- (e) انحراف استاندارد را از فرمول $(UNPL - A_m)/3$ محاسبه می کنیم

❖ **قواعد مرزی (زون) :** برای ارزیابی این مطلب استفاده شوند که آیا تغییرات نشان داده شده توسط این معیارها نشان دهنده ی فرآیندی هستند که تحت کنترل یا خارج از کنترل می باشد، اگر هریک از این ۴ اصل درست باشد نشان دهنده فرآیندی خارج از کنترل است:

• $S =$ انحراف استاندارد

- یک مقدار منفرد متریک ها خارج از UNPL
- از سه مقدار متوالی متریکها دوتا، $2S$ از A_m فاصله داشته باشند
- از پنج مقدار متوالی متریکها سه تا، S از A_m فاصله داشته باشند
- قرار داشتن هشت مقدار متوالی متریک ها، در یک سوی مقدار متوسط

متریکهایی برای سازمانهای کوچک

روش پیشنهاد شده برای پیشنهاد متریک های نرم افزاری برای سازمانهای کوچک

۱. ساده نگهداشتن: راهنمایی است که در حد معقولی در بسیاری از فعالیتها خوب کار می کند
۲. برطرف سازی نیازهای محلی
۳. اطمینان از افزایش ارزش

مجموعه اندازه های ساده برای سازمان نرم افزاری کوچک

- زمانی که سپری می شود (به ساعت یا روز) تا یک درخواست داده شده ، کاملاً ارزیابی شود. t_{queue}
- نیروی کار مورد نیاز (نفر-ساعت) برای ارزیابی W_{eval}
- زمانی که سپری می شود تا سفارش تغییر یک ارزیابی کامل شده به پرسنل داده شود. t_{eval}
- نیروی مورد نیاز برای اعمال تغییرات. W_{change}
- زمان مورد نیاز جهت اعمال تغییرات. t_{change}
- خطاهای یافت شده طی فعالیت اعمال تغییرات. E_{change}
- عیوب کشف شده پس از آنکه تغییرات اعمال و محصول به مشتری تحویل گردیده D_{change} و استفاده از این اندازه ها برای ایجاد متریک ها

$$DRE = E_{change} / (E_{change} + D_{change})$$

ایجاد برنامه ای برای متریکهای نرم افزار

۱. اهداف تجاری خود را تعریف می کنیم
۲. مشخص نمایید که چه چیزهایی را می خواهید بدانید
۳. اهداف فرعی خود را مشخص کنید
۴. موجودیتها و صفات خاصه مرتبط با اهداف فرعی را تعریف کنید.
۵. اهداف اندازه گیری خود را رسم نمایید.
۶. پرسشهای کمی و شاخصهای مرتبط را مشخص کنید
۷. عناصر داده ای را مشخص کنید
۸. اندازه های مورد استفاده را مشخص کنید
۹. اقداماتی که برای پیاده سازی این اندازه گیری ها باید انجام دهید، مشخص کنید.
۱۰. یک طرح و برنامه برای پیاده سازی اندازه گیری ها آماده کنید

برای بدست آوردن اهداف تجاری ، ممکن است نیاز به اولویت بندی داشته باشیم .

برای بدست آوردن بقیه سوالات ، يك لیست سوالات ایجاد می کنیم مثلا لیست سوالات موجودیت ایجاد می کنیم . این سوالات موجودیت به ما در شناسایی اهداف فرعی کمک می کند.

فصل پنجم : طرح ریزی پروژه نرم افزاری

برنامه ریزی پروژه نرم افزاری چیست؟

مدیریت پروژه نرم افزاری با مجموعه فعالیتهایی آغاز می شود که جمعاً **برنامه ریزی پروژه** یا project planing نامیده می شود. قبل از آغاز پروژه، مدیر و تیم نرم افزاری باید کاری را که قرار است انجام گیرد، منابع لازم و زمان لازم از شروع تا انتها را تخمین بزنند. شامل فعالیتهای مختلفی چون تخمین، تحلیل و مدیریت ریسک، زمانبندی و پیگیری پروژه، تضمین کیفیت و ... می باشد در این فصل، برنامه ریزی شامل تخمین می باشد یعنی **تلاش شما برای تعیین میزان پول، کار لازم، تعداد منابع و مقدار زمان** لازم برای ایجاد یک سیستم یا محصول خاص مبتنی بر نرم افزار.

□ چه کسی این کار (تخمین) را انجام می دهد؟

مدیران نرم افزاری با استفاده از اطلاعات جمع آوری شده از مشتریان و مهندسان نرم افزاری و داده های متریک نرم افزاری حاصل از پروژه های گذشته این کار را انجام می دهند.

□ دلیل اهمیت تخمین چیست؟

شما هیچگاه خانه ای را بدون برآورد هزینه نمی سازید منطقی است قبل از شروع به ایجاد نرم افزار، **برآوردی از هزینه** انجام دهیم.

□ چگونگی اطمینان از صحت این مرحله؟

با داشتن تجربه و روش سیستماتیک تخمین صورت می گیرد، با تعیین پیچیدگی و ریسک نسبت به انتخاب صحیح دید مطمئن می شوید

□ مراحل انجام تخمین؟

تخمین با توصیف دامنه محصول شروع می شود. تا وقتی که دامنه محدود است، توسعه ی تخمین معنی دار، امکانپذیر نیست سپس هر مسله به مسائل کوچکتر تجزیه می شود و هر یک توسط داده ها سابقه و تجربه تخمین زده می شوند توصیه که تخمین خود را حداقل با دو روش متفاوت انجام دهید. پیچیدگی مساله و خطر قبل از برآورد نهایی در نظر گرفته می شوند.

□ محصول نهایی چیست؟

جدول ساده ای که جزییات کارهایی را که باید صورت بگیرند، توابعی که باید ایجاد شوند، **و هزینه، تلاش، زمان** لازم برای هر یک را نشان می دهد، فهرستی از منابع لازم نیز ارائه می گردد.

مشاهده برآوردها (حقایقی در مورد برآوردها)

- ❖ کار تخمین بطور بالفطره دارای خطرانی (ریسکهای) هست و این خطر (ریسکها) است که منجر به عدم قطعیت می شود.
۱. پیچیدگی پروژه تاثیر شدیدی روی عدم قطعیت در برنامه ریزی دارد. پیچیدگی و سخت بودن کار یک معیار (اندازه) نسبی است که تحت تاثیر آشنایی با کار و تجربه گذشته است.
- تولید کننده یک برنامه کاربردی پیچیده تجارت الکترونیکی که اولین بار آن را انجام می دهد باید آنرا بسیار مشکل بداند، اما تیم نرم افزاری که این کار را برای ده بار انجام داده ، آن را بسیار سهل می داند.
 - برآوردهای در مورد کد ها و طراحی دارای پیچیدگی بالا هستند
۲. اندازه بزرگی پروژه عامل مهمی دیگری است که می تواند بر دقت و کارایی تخمین ها تاثیر بگذارد. با افزایش اندازه بزرگی وابستگی درونی میان عناصر مختلف نرم افزار به سرعت رشد می کنند. تجزیه مسئله که روش مهم دیگری برای تخمین زدن است ، سخت تر می باشد زیرا عناصر تجزیه شده ممکن است سخت تر و مشکل تر باشند.
۳. میزان عدم قطعیت ساختاری (عدم تعریف دقیق دامنه) نیز روی تخمین خطر تاثیر دارد. در اینجا ، ساختار اشاره دارد به درجه ای که نسبت به آن شرایط سخت شده ، میزان راحتی که با آن میتوان توابع و عملکردها را بخش بندی ، نمود و ماهیت سلسله مراتبی اطلاعاتی که باید پردازش شوند.
۴. در دسترس بودن اطلاعات تاریخی و سوابق قبلی دارای تاثیر مثبت شدیدی بر تخمین خطر (ریسک) است.
- ❖ خطر با میزان عدم قطعیت در انجام برآوردهای کیفی برای منابع، هزینه و جداول زمانبندی سنجیده میشود. اگر دامنه پروژه ای به خوبی شناخته نشده یا نیازمندیهای آن در معرض تغییر باشند ریسک و خطر به شدت بالا میرود.
- ❖ مدیر پروژه نباید در مورد تخمین وسواس خرج دهد. شیوه های مدرن مهندسی نرم افزار (مثل مدلهای فرآیند تکمیلی) دیدگاه تکراری از تولید دارند. در چنین روشهایی ممکن است میزان برآورد را مجدداً بازبینی نمود و و وقتی مشتری در نیازمندیها تغییراتی ایجاد می کند آنرا اصلاح کنیم.



اهداف اصلی طرح ریزی پروژه

هدف برنامه ریزی پروژه نرم افزاری عبارتست از **مهیا کردن چارچوبی** که مدیر را قادر به **ارائه تخمین منطقی** از منابع (نیازها) ، هزینه (پول\$\$\$) و زمانبندی (زمان انجام فعالیتها و ...) کند. تخمین ها در يك چارچوب زمانی در آغاز پروژه صورت گرفته و مرتباً با پیشروی پروژه بروز میشوند.

تخمین ها تلاش دارند که **بهترین و بدترین طرح ها** را معین کنند تا **نتیجه پروژه** را بتوان **محدود** نمود. اهداف برنامه ریزی از طریق **فرآیند کشف اطلاعاتی** حاصل میگردد که منجر به تخمین های منطقی میشود.

دامنه نرم افزار

اولین کار در برنامه ریزی پروژه عبارتست از تعیین دامنه آن.

عملکرد و کارایی مربوط به نرم افزار در مهندسی سیستم باید ارزیابی گردد که غیر مبهم و در سطوح مدیریتی و فنی **قابل درک** باشد. گزارشی از این دامنه باید پیوست شود.

دامنه نرم افزار **توصیف کننده اطلاعات و کنترل مورد پردازش**، عملکرد، کارایی، محدودیتها (شرایط حد)، رابط ها و قابلیت اطمینان (اگر درست عمل نکرد بازهم به بحران نرسیم) است

• محدودیتها در نرم افزار را بوسیله سخت افزار خارجی یا حافظه موجود یا دیگر سیستمهای موجود ایجاد می شوند

به دست آوردن اطلاعات مورد نیاز برای تعیین دامنه

□ **برقراری ارتباط بین مشتری و تولیدکننده**

رایج ترین تکنیک مورد استفاده برای برقراری ارتباط بین مشتری و تولیدکننده ، عبارتست از برگزاری **يك جلسه یا مصاحبه**.

۱- اولین مجموعه سئوالات آزاد روی **مشتری؛ اهداف کلی و فواید** تاکید دارد

مثلاً : چه کسی در پشت تقاضای این کار است؟ چه کسی از این راه حل استفاده میکند؟

حسن اقتصادی حل مساله چیست؟ آیا منبع دیگری برای حل وجود دارد؟ و

۲-سئوالات دوم تحلیلگر را قادر به شناخت بهتری از مسئله و مشتری میکند. مثلاً

این راه حل برای چه مشکلاتی است؟ آیا محیط عملیاتی با این راه حل وجود دارد؟

۳-سئوالات آخر(سوم) بر میزان موثر بودن جلسه تاکید دارد و آن را فوق پرسش مینامند و فهرست زیر پیشنهاد میشود :

آیا من سئوالات زیادی پرسیدم؟ آیا سئوالات من با مسئله ای که مورد حل قرار داده اید مرتبط اند؟...

این سئوالات (و سئوالات دیگر) کمی به تحرك اولیه مسئله(؟؟؟)(ترجمه بهتر: شکستن سردی بین افراد) کمک کرده و ارتباطی را که برای ایجاد دامنه پروژه لازم است ایجاد میکند.

❖ بخش سئوال و جواب تنها باید برای اولین برخورد استفاده شود و سپس در قالب جلسه ای جایگزین آن شود. که عناصر حل مسئله مذاکره و مشخصات را مرتبط میسازد.

❖ FAST^۱ : (ایجاد تیم مشترک به جای یادداشت و رابطه ما و آنها) = پروژه ای است با نام فنون ساده شده مشخصات برنامه های کاربردی که ایجاد تیم مشترکی از مشتریان و تولید کنندگانی که برای حل مسئله با هم کار میکنند را تشویق نموده، عناصر راه حل را پیشنهاد کرده و مجموعه ای از نیازمندیها را مشخص میکند.

امکان سنجی (مرحله تکمیلی بعد از تعیین دامنه)

• قدم بعد از تعیین دامنه (Scope) ، امکان سنجی است

• وقتی دامنه شناسایی شد منطقی است بپرسیم که آیا این پروژه شدنی است؟

❖ عملی بودن نرم افزار (امکان سنجی) دارای چهار بعد است

۱. فن آوری : آیا از لحاظ تکنیکی قادر به انجام پروژه هستیم؟

۲. هزینه: آیا با هزینه های صورت گرفته امکان انجام پروژه هست؟

۳. زمان: آیا زمان تولید تا رسیدن پروژه به بازار ، قابل قبول است؟

۴. منابع: آیا سازمان تمام منابع مورد نیاز را در اختیار تیم قرار داده است؟

تعیین دامنه کافی نیست. وقتی دامنه فهمیده شد ، تیم نرم افزاری و دیگران باید مشخص سازند که با ابعاد روشن شده آیا امکان کار وجود دارد یا خیر ؟ که این خود قسمتی از فرآیند تخمین و برآورد خواهد بود

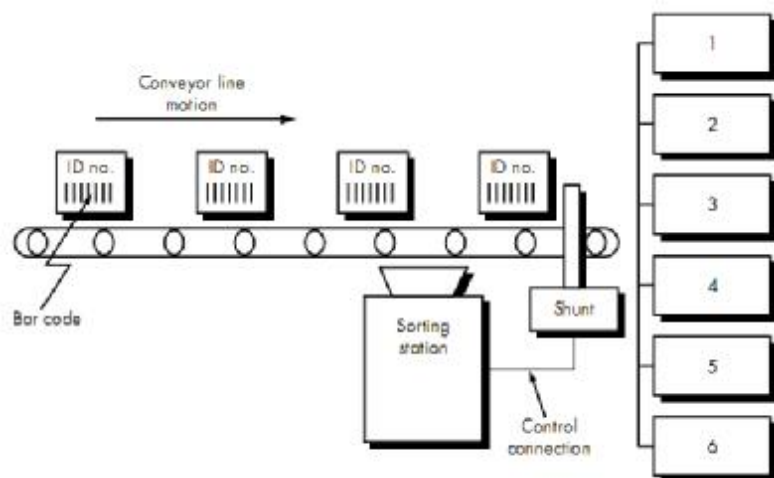
مثال

مثال / نرم افزاری را برای نوعی سیستم مرتب کردن خط حمل و نقل یا CLSS در نظر بگیرید وضعیت دامنه را برای آن مشخص کنید.

[برای اطلاعات بیشتر رجوع شود به صفحه ۱۶۰]

• تجزیه (طبق فصل ۳)

- خواندن ورودی بار کد.
- خواندن پالسهای دورسنج.
- دکد نمودن داده های کد شده.
- تعیین مکان قرارگیری جعبه.
- تولید سیگنال کنترلی برای اهرم.
- نگهداری اطلاعات مقصد جعبه ها.



دامنه نرم افزار توصیف کننده اطلاعات و کنترل مورد پردازش ، عملکرد، کارایی، محدودیتها (شرایط حد)، رابط ها و قابلیت اطمینان (اگر درست عمل نکرد بازهم به بحران نرسیم) است

- عملکرد را کارایی مشخص می کند
- مثال کارایی: امکان تغییر سرعت از ۵ به ۱۰
- مثال محدودیت: جعبه ها از هم فاصله نداشته باشند
- مثال رابطه: با سخت افزار + نرم افزار مثلا: بانک اطلاعات ذرجه قطعات + افراد + رویه های قبل از بعد نرم افزار
- قابلیت اطمینان : قطعه بی نشان را نگه ندارد چون خطا ایجاد می شود

منابع

دومین کار در برنامه ریزی نرم افزاری تخمین منابع لازم برای نیل به کار تولید نرم افزار است.

□ منابع مورد نیاز برای تولید نرم افزار :

۱- محیط توسعه: شامل ابزارهای نرم افزاری و سخت افزاری و به عنوان زیربنای برای حمایت از تلاشهای توسعه است

۲- منابع نرم افزاری با قابلیت استفاده مجدد: که سبب پایین آوردن هزینه تولید و سرعت بخشیدن به تحویل نرم افزار

۳- منابع اولیه: شامل مردم می باشد در نوک هرم قرار گرفته اند
□ هر منبع دارای چهار مشخصه است:

• توصیف منبع ، وضعیت دسترسی به آن، زمان تقویمی که در آن موقع مورد نیاز است، مدت زمانی که منبع بکار گرفته میشود.

• دو مشخصه زمان تقویمی که در آن موقع مورد نیاز است، مدت زمانی که منبع بکار گرفته میشود را میتوان بعنوان يك پنجره در نظر گرفت.

• در دسترس بودن منبع برای يك پنجره بخصوص باید در اولین زمان ممکن صورت گیرد.

۱- منابع انسانی

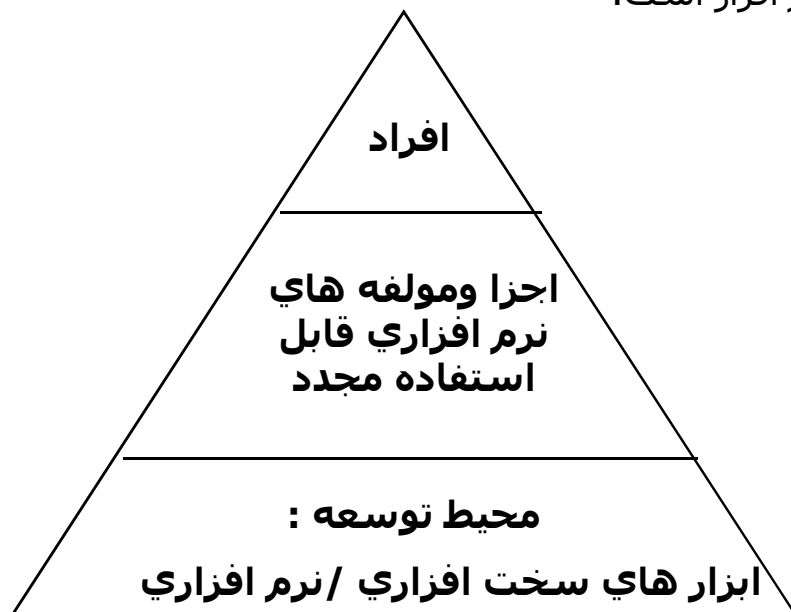
۲- منابع نرم افزاری قابل استفاده مجدد

۳- منابع محیطی

• منابع

منابع انسانی

تعداد افراد لازم برای پروژه را می توان تنها بعد از تخمین کار تولید، ارائه کرد.



منابع نرم افزاری با قابلیت استفاده مجدد:

مهندسی نرم افزار بر مبنای اجزاء (CBSE) بر فعالیت کاربرد مجدد تاکید دارد یعنی ایجاد و استفاده مجدد بلوکهای ساختمان نرم افزار: چنین بلوکهایی به نام جزء یا اجزا نامیده می شوند اجزا برای ارجاع آسان باید طبقه بندی شوند، برای به کارگیری آسان باید استاندارد شوند و برای ترکیب ساده باید ارزیابی شوند.

□ چهار گروه نرم افزاری را باید در برنامه ریزی مد نظر قرار گیرند

- ۱- اجزای ساخته شده حاضر و آماده (COTS): آنها را آماده می خریم و آماده و ارزیابی شده اند
- ۲- اجزایی که دارای تجربه کاملی در مورد آنها هستیم: اجزای پروژه های قبلی که مشابه نرم افزار فعلی اند و افراد تیم جاری کاملاً با اجزای قبلی آشنا هستند و نیازمند پاره ای از اصلاحات هستند و کم خطر (ریسک) اند.
- ۳- اجزایی که دارای تجربه نسبی در مورد آنها هستیم: اجزای پروژه های قبلی که مشابه نرم افزار فعلی اند و افراد تیم جاری با اجزای قبلی آشنایی محدودی دارند و نیازمند پاره ای از اصلاحات هستند لذا درجه پایینی از ریسک وجود دارد
- ۴- اجزای جدید: اجزایی که توسط تیم نرم افزار بایست ایجاد شوند.

□ هنگامی که اجزای قابل استفاده مجدد بعنوان يك منبع استفاده میشوند رهنمودهای زیر باید مد نظر باشند:

۱. اگر اجزای آماده مصرف (خارجی) نیازهای پروژه را برآورد میکنند آنها را بخرند.
۲. اگر اجزایی که در مورد آنها تجربه کامل داریم در دسترس هستند خطر مربوط به اصلاح و الحاق آنها معمولاً پذیرفتنی است. و استفاده مشکلی ندارد
۳. اگر تجربه نسبی از این جز داریم استفاده آن برای پروژه کنونی باید تحلیل شود.

منابع محیطی

- محیطی که پروژه نرم افزاری را پشتیبانی میکند ، محیط مهندسی نرم افزار (SEE) است به کار گیرنده سخت افزار و نرم افزار است.
- سخت افزار سکویی است که ابزارهای لازم (نرم افزاری) برای تولید محصولات کاری را تامین میکند که نتیجه يك کار مهندسی خوب است.
- برنامه ریز پروژه باید پنجره زمانی لازم برای سخت افزار نرم افزار را مشخص کند

برآورد پروژه های نرم افزاری

• امروزه تخمینهای هزینه و فعالیت نرم افزار به دلیل پیچیده شدن و گران شدن نرم افزار از يك هنر ناشناخته به يك سري مراحل سیستماتيك تبدیل شده است

□ برای رسیدن به تخمین های مورد اطمینانی از نظر هزینه و کار يك سري گزینه وجود دارد:

۱. کار تخمین را تا اواخر پروژه به تعویق بیندازیم. (واضح است تخمین بعد از پروژه ۱۰۰٪ درست است) (برای پروژه های بعدی)
۲. برآوردها را بر پایه پروژه هایی قرار دهیم که هم اکنون تکمیل شدند.
۳. از فنون نسبتاً ساده تفکیک و تجزیه برای ارائه این برآوردها استفاده کنیم. (تخمین زیرمستله ها...)
۴. از يك یا چند مدل تجربی برای آن استفاده نماییم.

روش اول چندان عملی نیست - روش دوم در بسیاری موارد خوب است ولی نه همیشه

روش تجزیه از رهیافت تقسیم و غلبه استفاده می کند، با تفکیک پروژه بصورت توابع و فعالیتهای مهندسی مربوطه تخمین هزینه و کار بصورت مرحله ای انجام میگیرد.

مدل تجربی برآورد را، میتوان برای فنون جدا سازی تکمیلی استفاده کرد و روش تخمین ارزشمندی را در جای درست خود ارائه داد.

مدل تجربی به صورت $d = f(v_i)$ تعریف می شود که d یکی از مقادیر تخمین (زمان، هزینه) را شامل می شود و v_i پارامتر مستقل انتخابی (مثل fp تخمینی یا loc تخمینی)

نکته: ابزارهای خودکار تخمین برآورد يك یا چند تکنیک تجزیه یا مدلهای تجربی را به اجرا در می آورند وقتی این ابزارها با رابط گرافیکی کاربر ترکیب میشوند گزینه جالبی برای انجام تخمین ارائه می دهند. در این سیستمها مشخصه های سازمان تولیدی (مثل تجربه و محیط) و نرم افزاری که قرار است تولید شود توصیف میگردند و تخمین از داده ها بدست می آید بدون داده های سابقه، برآورد هزینه اساس متزلزلی خواهد داشت.

فنون تجزیه

در اکثر موارد ارائه برآورد هزینه و فعالیت بزرگتر از حدی است که به صورت یک قطعه در نظر گرفته شود، در نتیجه برنامه به قطعات کوچکتری تجزیه می شود در فصل ۳ تجزیه را از دو دیدگاه : تجزیه مسئله و تجزیه فرآیند بررسی نمودیم در تخمین از یکی یا هر دو شکل تجزیه استفاده می کنیم. قبل از انجام برآورد (تخمین) برنامه ریز، پروژه باید دامنه نرم افزاری را که قرار است ساخته شود شناخته و تخمینی از اندازه آن ارائه کند.

□ **میزان دقت تخمین پروژه بر اساس چند چیز پیش بینی میشود (اصول صحت تخمین):**

- ۱- **درجه درستی برآورد اندازه محصول:** درجه ای که نسبت به آن برنامه ریز به درستی اندازه محصول کاری را تخمین زده است.
- ۲- **توانایی ترجمه،** تخمین اندازه نیروی کار انسانی، تقویم زمانی و میزان پول.
- ۳- **درجه انعکاس توانایی های تیم نرم افزاری:** درجه که نسبت به آن طرح پروژه منعکس کننده تواناییهای تیم نرم افزاریست.
- ۴- **پایداری خواسته های پروژه و محیط:** ثبات نیازمندیهای محصول و محیطی که کار مهندسی نرم افزار را پشتیبانی میکند.

تعیین اندازه نرم افزار

تخمین اندازه نشان دهنده اولین کوشش عمده برنامه ریز پروژه می باشد، اندازه به نتیجه کمی پروژه اشاره دارد و اگر روش مستقیم به کار گرفته شود، اندازه می تواند بر حسب LOC و اگر روش غیر مستقیم به کار گرفته شود اندازه بر حسب FP نشان داده می شود

□ **چهار روش مختلف برای تعیین میزان مسئله :**

- ۱- **شیوه اندازه زدن با منطق فازی:** بیان اندازه بر مبنای مقیاس کمی و پالایش آن در محدوده اصلی
- ۲- **شیوه اندازه زدن با امتیازات عملکردی:** فصل ۴ و بحث FP
- ۳- **شیوه اندازه زدن با جزء استاندارد:** تخمین تعداد وقوع هر جز استاندارد (مثل: صفحات-گزارشها-فایلها-LOC و...) و استفاده از داده های سابقه برای تخمین اندازه نهایی محصول
- ۴- **شیوه اندازه زدن با تغییر اندازه:** زمان اصلاح نرم افزار استفاده می شود

۱- برآورد مبتنی بر مساله

داده های بدست آمده از روشهای خطوط کد (LOC) و امتیازات کاربردی (FP) که از روی آنها می توان متریک را حساب کرد علاوه بر استفاده ای که در ارزیابی کیفیت داشتند در طول کار پروژه به دو صورت مورد استفاده قرار میگیرند:

۱- به عنوان متغیر تخمینی که برای تعیین اندازه هر عنصر نرم افزاری استفاده می شود.

۲- به عنوان متریک خط مبنای که از روی پروژه های قبلی جمع آوری شده همراه با متغیرهای تخمینی برای ارائه هزینه و میزان نیروی کار پروژه استفاده می شود.

• برآورد FP و LOC فنون متمایزی از تخمین هستند. هر دو دارای چند مشخصه مشترک هستند .

• مراحل تخمین کمی در مورد پروژه:

۱. برنامه ریز پروژه با وضعیت محدود شده دامنه نرم افزار شروع کرده و از روی آن تلاش می کند نرم افزار را به صورت توابعی از مساله در بیاورد که بتوان هر کدام را به صورت مجزا برآورد کرد. FP و LOC برای هر تابعی تخمین زده می شود. یا برنامه ریز جز دیگری را برای تعیین اندازه انتخاب می کند مثل کلاسها یا اشیا، تغییرات یا فرایندهای تجاری تحت تاثیر قرار گرفته

۲. سپس متریک های بهره وری خط مبدا مانند (LOC/pm یا FP/pm-مخفف نفر-ماه) با متغیر برآورد مناسب بکار گرفته شده و هزینه یا نیروی کار تابع به دست می آید

۳. از روی تخمین توابع، تخمین کمی در مورد کل پروژه بدست می آید.

• نکته مهم مورد توجه این است که اغلب نکات پراکنده عمده ای در متریک های بهره وری یک سازمان وجود دارد که باعث مشکوک شدن استفاده از یک متریک تولید مبنا می شوند

• فنون تخمین LOC و FP از نظر جزئیات لازم برای تفکیک و هدف تقسیم بندی با هم متفاوتند

• وقتی LOC به عنوان متغیر تخمین استفاده می شود تجزیه کاملاً ضروری است و اغلب از نظر جزئیات بررسی می شود. و بر روی توابع نرم افزار متمرکز می شویم

• در مورد تخمین های FP کار جدا سازی به صورت دیگری است علاوه بر تمرکز تابع هر یک از مشخصه های دامنه اطلاعات یعنی ورودی ها خروجی ها فایل های اطلاعاتی و... به علاوه ۱۴ مقدار تطابق پیچیدگی (f_i ها) برآورده می شود.

□ مراحل تخمین

• برنامه ریز بدون توجه به متغیر تخمین با برآورد یک سری مقادیر برای هر یک از تابع ها یا مقدار دامنه اطلاعاتی شروع می کند، با استفاده از اطلاعات تاریخی یا حدس برنامه ریز اندازه بزرگی هر تابع را به صورت خوشبینانه یا از روی بد بینی تخمین می زند. میزان عدم قطعیت هنگامی مهیا می شود که دامنه ای از اعداد مشخص شوند.

• سپس یک مقدار مورد انتظار یا سه امتیازی محاسبه می شود.

• مقدار منتظره برای متغیر تخمینی (اندازه) S را می توان به عنوان یک میانگین ارزیابی شده از تخمین های خوشینانه (S_{opt})، دارای احتمال بالا (S_m) و تخمین بدبینانه (S_{pess}) ارزیابی کرد. به طور مثال:

•Pessimistic

•most likely

•optimistic

$$S = (S_{opt} + 4S_m + S_{pess}) / 6$$

• بیشترین اعتبار را به تخمینی که احتمال زیادی دارد داده و توزیع احتمال بتا را دنبال می کند.

• وقتی مقدار مورد انتظار متغیر تخمین معین شد FP و LOC به کار گرفته می شوند .

• آیا این تخمین ها درست هستند؟

نمی توان مطمئن بود



مثال ۱: برآورد مبتنی بر خطوط برنامه (LOC)

۱. وضعیت اولیه دامنه (نامحدود)

- به عنوان نمونه ای از فنون تخمین LOC و FP, بسته نرم افزار باید در نظر بگیرید که قرار است برای یک برنامه کاربردی طراحی با کمک کامپیوتر (CAD) برای اجزای مکانیکی تولید شود.
- بازنگری مشخصات سیستم نشانگر این است که نرم افزار قرار است روی یک ایستگاه مهندسی اجرا شود و باید با محیطهای گرافیکی مختلف کامپیوتری از جمله موس , دیجیتایزر, صفحه نمایش دارای تفکیک پذیری بالا و چاپگر لیزری ارتباط برقرار کند.
- نرم افزار CAD اطلاعات هندسی دو و سه بعدی مهندس را دریافت می کند.

۲. سپس اقدامات اولیه را بسط داده تا جزئیات اساسی و محدودیتهای کمی ایجاد شوند

- مهندس با سیستم ارتباط برقرار کرده و CAD را از طریق رابط کاربردی کنترل می کند که مشخصات یک طراحی رابط متقابل ماشین - انسان را به خوبی نشان میدهد . یا اینکه اندازه و میزان پیچیدگی پایگاه داده ای CAD باید چه باشد.
- ۳. به منظور دستیابی به اهداف مدنظر , ما فرض می کنیم که پالایش بیشتری رخ داده و **عملکردهای** نرم افزاری زیر **شناسایی** شده است :

- ۱- تسهیلات کنترلی و رابط کاربر (UICF)
- ۲- تحلیل هندسی دو بعدی (2DGA) ۳- تحلیل هندسی سه بعدی (3DGA)
- ۴- مدیریت پایگاه اطلاعاتی (DBM)
- ۵- تسهیلات نمایشی گرافیک کامپیوتر (CGDF)
- ۶- کارکرد کنترل دستگاه های جانبی (PCF)
- ۷- پیمانه های تحلیل طراحی (DAM)

تخمین LOC	فانکشن
۲۳۰۰	تسهیلات کنترلی و رابط کاربر (UICF)
۵۳۰۰	تحلیل هندسی دو بعدی (2DGA)
۶۸۰۰	تحلیل هندسی سه بعدی (3DGA)
۳۳۵۰	مدیریت پایگاه اطلاعاتی (DBM)
۴۹۵۰	تسهیلات نمایشی گرافیکی (CGDF)
۲۱۰۰	کارکرد کنترل دستگاه های جانبی (PCF)
۸۴۰۰	پیمانه های تحلیل طراحی (DAM)
۳۳۳۰۰	تعداد خطوط برآورد شده برنامه

مقادیر حوزه اطلاعاتی	خوش بینانه	متوسط	بد بینانه	شمارش برآورد	وزن	امتیاز عملیاتی
تعداد ورودی ها	۲۰	۲۴	۳۰	۲۴	۴	۹۷
تعداد خروجی ها	۱۲	۱۵	۲۲	۱۶	۵	۷۸
تعداد پرس و جو ها	۱۶	۲۲	۲۸	۲۲	۴	۸۸
تعداد پرونده ها	۴	۴	۵	۴	۱۰	۴۲
تعداد رابط های خارجی	۲	۲	۳	۲	۷	۱۵
تعداد کل						۳۲۰

$$FP_{estimated} = \text{count-total} \times [0.65 + 0.01 \times \sum (F_i)] = 375$$

۴- ایجاد جدول برآورد و تخمین برای روش تعداد خطوط برنامه

$$S = (S_{opt} + (4S_m + S_{pess})/6) : \text{با استفاده از معادله}$$

ستون تخمین LOC را ایجاد می کنیم

برای تابع هندسی سه بعدی ، بهترین حالت ۴۶۰۰ ، در محتملترین حالت ۶۹۰۰ و در بدترین حالت ۸۶۰۰ مقدار منتظره برای تابع هندسی سه بعدی ۶۸۰۰ می شود . LOC سایر تخمین ها به شیوه ای مشابه مشتق می شود . با جمع بندی در ستون LOC تخمینی برآورد ۳۳۱۵۰ خط کد برای سیستم ADC ارائه می شود.

طبقه ، سابقه سازمانی تولید 620loc/pm پس ۵۴ نفر ماه

مثال ۲: برآورد مبتنی بر امتیاز کارکردی (FP)

• تفکیک سازی در مورد تخمین مبتنی بر FP علاوه بر کارکردهای نرم افزاری روی مقادیر دامنه اطلاعات نیز متمرکز می شود.

• بایادآوری جدول محاسبه امتیازات عملکردی ، برنامه ریز پروژه ، ورودی ها ، خروجی ها ، درخواستها ، فایلها و رابطه های خارجی را در مورد نرم افزار CAD تخمین می زند.

• در مورد اهداف این تخمین ، عامل ارزیابی پیچیدگی به طور متوسط در نظر گرفته می شود .

• نهایتا میزان تخمین زده شده FP به دست می آید

• طبق سابقه سازمانی تولید 6.5fp/pm پس ۵۸ نفر ماه

۲- برآورد مبتنی بر فرایند

- ❖ رایج ترین تکنیک تخمین زدن یک پروژه ، بر مبنای فرآیندی است که استفاده می شود. یعنی فرآیند به چند مجموعه وظایف نسبتاً کوچک تقسیم می شود و نیروی کار لازم برای هر فرآیند تخمین زده می شود.
- ❖ مانند روشهای مبتنی بر مبنای مسئله ، این تخمین هم با نمایش وظایف بدست می آید.
- ❖ وقتی کارکردها و فعالتهای فرایندی اعلام شدند برنامه ریز نیروی کاری را که (مثلاً نفر-ماه) برای رسیدن به فرایند هر کارکرد نرم افزاری لازم است برآوردمی کند میزان متوسط نیروی کار (هزینه/واحد نیروی کار) در نیروی کار تخمین زده شده برای هر فرایند به کار گرفته می شود. این احتمال وجود دارد که میزان نیروی کار در هر مورد متفاوت باشد.
- ❖ کارکنان ارشد شدیداً در فعالتهای اولیه دخیلند و معمولاً از کارکنان سطح پایین تر که در کارهای طراحی ، ایجاد کد و آزمون اولیه دخالت دارند گرانتر می باشند.
- ❖ هزینه و نیروی کار برای هر عملیات و فرایند نرم افزاری به عنوان آخرین مرحله استفاده می شود.

مهندسی	تحلیل ریسک	طرح ریزی	ارتباط با مشتری	فعالتهای چارچوب مشترک فرایند
				فعالتهای مهندسی نرم افزار
				کارکردهای محصول
				ورودی متن
				ویرایش و قالب بندی
				اصلاح خودکار
				آرایش صفحه و صفحه بندی
				شاخص بندی خودکار و Toc
				مدیریت فایل
				تولید مستندات

مثال : برآورد مبتنی بر فرایند

- به منظور تشریح استفاده از تخمین مبتنی بر فرآیند نرم افزار CAD را در نظر می گیریم (مانند مثالهای قبل)
- بر اساس میانگین نیروی کار به منظور ۵۰۰۰ دلار در ماه، هزینه تخمین زده کل پروژه ۲۳۰۰۰۰ دلار بوده و نیروی کار تخمینی نفر در ماه میباشد. اگر بخواهید میزان نیروی کار با هر فرایند نرم افزاری یا مهندسی مرتبط شده و جداگانه محاسبه می شود.
- نیروی کار کل تخمینی در مورد دامنه های نرم افزاری CAD از حداقل ۴۶ نفر در ماه (روش فرایند) تا حداکثر ۵۸ نفر (روش fp) میرسد. میانگین برآورد از کل سه روش ۵۳ نفر - ماه است. حداکثر **میزان تغییر از متوسط تخمین زده شده** تقریباً ۱۲ درصد است.
- به خاطر اینکه میزان نیاز به متخصص هر فعالیت فرایند جداگانه حساب می شود، هزینه بر اساس کار دقیقتر محاسبه می شود
- وقتی میزان سازگاری بین تخمین ها ناچیز است چه اتفاقی رخ داده؟ ارزیابی مجدد اطلاعات

Activity →	CC	Planning	Risk analysis	Engineering		Construction release		CE	Totals
Task →				Analysis	Design	Code	Test		
Function ↓									
UICF				0.50	2.50	0.40	5.00	n/a	8.40
2DGA				0.75	4.00	0.60	2.00	n/a	7.35
3DGA				0.50	4.00	1.00	3.00	n/a	8.50
CGDF				0.50	3.00	1.00	1.50	n/a	6.00
DBM				0.50	3.00	0.75	1.50	n/a	5.75
PCF				0.25	2.00	0.50	1.50	n/a	4.25
DAM				0.50	2.00	0.50	2.00	n/a	5.00
Totals	0.25	0.25	0.25	3.50	20.50	4.50	16.50		46.00
% effort	1%	1%	1%	8%	45%	10%	36%		

CC = customer communication CE = customer evaluation

- تخمین های بسیار متفاوت را اغلب به یکی از دلایل زیر بدست می آیند

- ۱- دامنه پروژه به اندازه کافی شناخته نشده یا برنامه ریز برداشت نادرستی از آن داشته است
- ۲- اطلاعات بهره وری مورد استفاده برای فنون تخمین مبتنی بر مساله، از نظر برنامه مناسب نیستند یا درست مورد استفاده قرار نگرفته اند.

ستونها: فعالیتهای فرایند
سطرها: تجزیه مساله (توابع)
خانه ها: هزینه هرتابع به ازای نوع فعالیت فرایند

PM (نفر)



مدل های برآورد تجربی

يك مدل تخمین برای نرم افزار کامپیوتر، از تجارب حاصل از فرمولها برای پیش بینی فعالیت به صورت تابعی از LOC با FP استفاده می شود، مقادیر LOC یا FP طبق روشهای برآورد گفته شده بدست می آیند

اطلاعات تجربی که اکثر مدلهای تخمینی را پشتیبانی می کنند از تعداد محدودی پروژه بدست می آیند. به ایندلیل هیچ مدل تخمینی برای تمام رده های نرم افزاری مناسب نیست و در همه محیطهای تولیدی مناسب نمی باشد. لذا نتایج این مدلها با تعدیل استفاده شوند

ساختار مدل های برآورد

یک مدل تخمینی عادی با استفاده از تحلیل بازگشتی یا رگرسیون بر روی اطلاعات جمع آوری شده از پروژه های قبلی، بدست می آید ساختار کلی چنین مدلهایی به شکل زیر است

$$E = A + BX(ev)^c$$

که در آن A, B, C ثابت بدست آمده به صورت تجربی هستند E نیروی کار برحسب نفر-ماه و ev متغیر تخمین (fp یا LOC) می باشد

$$\text{effort} = \text{tuning coefficient} * \text{size}^{\text{exponent}}$$

$$E = 5.2 \times (KLOC)^{0.91}$$

Walston-Felix model

$$E = 5.5 + 0.73 \times (KLOC)^{1.16}$$

Bailey-Basili model

$$E = 3.2 \times (KLOC)^{1.05}$$

Boehm simple model

$$E = 5.288 \times (KLOC)^{1.047}$$

Doty model for KLOC > 9

FP-oriented models have also been proposed. These include

$$E = -13.39 + 0.0545 FP$$

Albrecht and Gaffney model

$$E = 60.62 \times 7.728 \times 10^{-8} FP^3$$

Kemerer model

$$E = 585.7 + 15.12 FP$$

Matson, Barnett, and Mellichamp model



• باری بوهوم در کتاب خود "اقتصاد مهندسی نرم افزار" یک سری مدل‌های تخمین نرم افزار معرفی میکند که دارای نام کوکومو هستند (مدل هزینه ساختاری)

• مدل اصلی کوکومو از همه بیشتر مورد استفاده قرار گرفته و مدل‌های تخمین هزینه در صنعت مورد بحث قرار گرفتند.

• COCOMO II مدل تخمین تکامل یافته کوکومو بود که زمینه های زیر را مورد توجه قرار داد :

۱. مدل ترکیبی کاربردی:

این مدل در طول مراحل اولیه مهندسی نرم افزار به کار میرود. هنگامی که **نمونه های اولیه** رابط کاربر مدلسازی میشوند با در نظر گرفتن رابطه متقابل سیستم و نرم افزار **بر آورد عملکرد و ارزیابی تکامل فناوری** که دارای اهمیت هستند

۲. مدل مرحله اولیه طراحی

وقتی بکار میرود که نیاز مندیها مشخص و معماری مقدماتی نرم افزار ایجاد شده

۳. مدل مرحله آخر معماری

در طول ساخت نرم افزار بکار میرود

• مدل‌های کوکومو ۲، نیازمند اطلاعات اندازه گیری است. سه گزینه از نظر اندازه به عنوان بخشی از سلسله مراتب مدل مهیا شده اند
امتیاز شی (OP) و امتیاز کارکردی (FP) و خطوط کد منبع (LOC)

• امتیاز شی یک معیار نرم افزاری غیر مستقیم است که با استفاده از شمارش ۱- **تعداد صفات نمایش** (پنجره ها) ۲- تعداد **گزارشات و ۳- تعداد اجزایی که احتمالاً برای ساخت برنامه لازمند (3GL)** محاسبه میشود.

• هر مورد شی در یکی از سطوح پیچیدگی (ساده-متوسط-مشکل) گروهبندی میشود. در اصل پیچیدگی تابعی از ۱- تعداد و منبع جدول اطلاعات خادم/مخدوم است که برای تولید صفحه یا گزارش لازم بوده و ۲- تعداد دیدگاهها یا بخش ها بعنوان بخشی از صفحه یا گزارش ارایه شده اند.

• فرمولهای پیدا کردن تعداد

• "امتیاز جدید شی:"

درصد استفاده مجدد



$$NOP = (100 - \%reuse) / 100 \times (\text{امتیاز شی})$$

• برای بدست آوردن تخمین نیروی کار بر اساس مقدار بالا باید "میزان بهره وری" را بدست آورد:

$$PROD = NOP / PM (\text{نفر ماه})$$

• "تخمین نیروی کار پروژه:"

$$\text{estimated effort} = NOP / PROD$$

وزن پیچیدگی			نوع
مشکل	متوسط	ساده	
۳	۲	۱	نمایشگر
۸	۵	۲	گزارش
۱۰	-	-	مولفه

وزن پیچیدگی برای انواع اشیا

قابلیت / تجربه	خیلی کم	کم	متوسط	بالا	خیلی بالا
قابلیت / بلوغ محیط	خیلی کم	کم	متوسط	بالا	خیلی بالا
PROD (بهره وری)	۴	۷	۱۳	۲۵	۵۰

نرخ های بهره وری برای امتیازات اشیا (PROD)



معادله نرم افزار (فرمولها)

معادله نرم افزار یک مدل دینامیک چند متغیره است که توزیع نیروی کار معینی را در طول حیات پروژه تولید نرم افزار فرض میگیرد.

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

مدل تخمینی:

E = نیروی کاری فرد در ماه یا سال

T = مدت پروژه به سال یا ماه

B = عامل مهارت های ویژه

P = پارامتر بهره وری که منعکس کننده موارد زیر است

- تکامل کلی فرایند و شیوه های مدیریتی
- میزان و مقیاسی که نسبت به آن روشهای خوب مهندسی نرم افزاری استفاده میشوند.
- سطح زبانهای برنامه نویسی مورد استفاده.
- وضعیت محیط نرم افزاری.
- مهارتها و تجربه تیم نرم افزاری.
- پیچیدگی برنامه کاربردی.

• معادله نرم افزاری دارای دو پارامتر مستقل است:

۱- تخمینی از اندازه (LOC)

۲- نشانگری از مدت پروژه به ماهها یا سالهای تقویمی

• حد اقل زمان تولید

$$t_{\min} = 8.14 (LOC/P)^{0.43} \text{ in months for } t_{\min} > 6 \text{ months}$$

$$E = 180 B t^3 \text{ in person-months for } E \geq 20 \text{ person-months}$$

تصمیم گیری ساخت / خرید

در حوزه های متعددی از برنامه های کاربردی مقرون به صرفه تر است که بجای تولید نرم افزار کامپیوتر آن را خریداری کنیم . مدیران مهندسی نرم افزار با تصمیم گیری در **خرید یا تولید** مواجه هستند گزینه های در دسترس:

۱- ممکن است نرم افزار بصورت آماده خریداری شود.

۲- اجزای نرم افزاری بصورت "با تجربه کامل" یا "با تجربه نسبی" بدست آیند و سپس اصلاح شده و یکپارچه گردند تا نیازها را بر آورده سازند.

۳- ممکن است نرم افزار توسط پیمانکاری خارج از سازمان طبق سفارش ساخته شده باشد تا نیازهای خریدار را مرتفع سازد.

در مورد محصولات نرم افزاری گرانتر رهنمودهای زیر را میتوان بکار گرفت:

۱- مشخصاتی برای کارکرد و عملکرد نرم افزار مطلوب ارایه میدهد.

۲- هزینه اولیه تا تولید و تاریخ تحویل را میگوید.

۳- الف: سه یا چهار برنامه نمونه انتخاب میکند که به بهترین نحو با مشخصه های شما جور میشود.

ب : اجزای نرم افزاری قابل استفاده مجددی انتخاب میکند که در ساخت برنامه مورد نیاز شما را یاری میکنند.

۴- شبکه مقایسه ای تشکیل میدهد که نمایانگر مقایسه سر به سر عملیات اصلی است.

۵- هر بسته نرم افزاری یا مولفه را بر اساس کیفیت محصول گذشته پشتیبانی فروشنده راهنمای محصول و... ارزیابی میکند.

۶- با سایر کاربران نرم افزار تماس گرفته نظراتشان را جویا میشود

در مورد تصمیم نهایی خرید یا تولید بر مبنای شرایط زیر صورت می گیرد :

۱. آیا تاریخ تحویل محصول زودتر از تاریخ تولید داخلی است؟

۲. آیا هزینه سفارش و خرید ، کمتر از هزینه تولید است؟

۳. آیا هزینه پشتیبانی خارجی ، کمتر از هزینه پشتیبانی داخلی (تیم ما) است؟

ایجاد یک درخت تصمیم گیری

مراحل توصیف شده فوق را میتوان با استفاده از فنون آماری در تصمیم گیری (مثل درخت تصمیم گیری) ارزیابی کرد

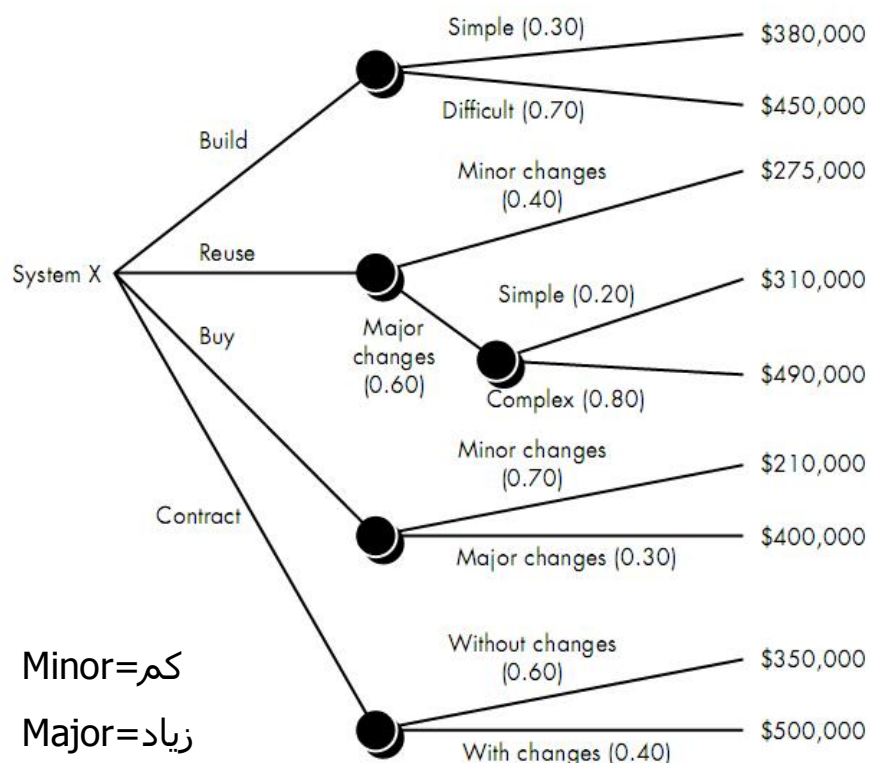
حالتهای درخت تصمیم گیری فرضی برای ساخت سیستم x

۱- ایجاد

۲- استفاده از مولفه های قبلی

۳- خرید و اصلاح نرم افزار

۴- قرارداد با سایر تیم ها



هزینه مورد نظر = (مسیر احتمالی) * (هزینه مسیر تخمینی)

هزینه های منتظره :

$$\text{expected cost}_{\text{reuse}} = 0.40 (\$275\text{K}) + 0.60 [0.20(\$310\text{K}) + 0.80(\$490\text{K})] = \$382\text{K}$$

$$\text{expected cost}_{\text{buy}} = 0.70(\$210\text{K}) + 0.30(\$400\text{K}) = \$267\text{K}$$

$$\text{expected cost}_{\text{contract}} = 0.60(\$350\text{K}) + 0.40(\$500\text{K}) = \$410\text{K}$$

کمترین هزینه ، هزینه خرید است



استفاده از منابع خارجی

❖ منابع خارجی از نظر مفهوم بسیار ساده هستند. کارهای طراحی نرم افزار با شخص ثالثی قرارداد بسته میشوند که این شخص کمترین هزینه و بالاترین کیفیت کار را انجام دهد.

❖ سطوح استفاده از منابع خارجی:

- ۱- سطح استراتژیک (راهبردی): مدیریت تجاری تصمیم می گیرد: آیا می توان بخش عمده ای از روند تولید را با عقد قرار داد انجام داد
 - ۲- سطح تاکتیکی: مدیر پروژه تصمیم می گیرد : آیا امکان انجام بخشی یا تمام کار توسط دیگران میسر است؟
- ❖ سود: معمولا پس اندازه های هزینه ای با کاهش تعداد افراد مربوطه به کار نرم افزاری حاصل میگردد که از آنها حمایت
- ❖ ضرر: شرکت کنترل نرم افزاری را که به آن نیاز دارد از دست میدهد.

ابزارهای خود کار بر آورده

گرچه ابزارهای تخمین خودکار بسیاری وجود دارند اما همه مشخصه های عمومی یکسانی نشان داده و همه شش عملکرد کلی زیر را اجرا میکنند:

- ۱- اندازه بندی موارد قابل تحویل پروژه.
- ۲- انتخاب فعالیتهای پروژه.
- ۳- پیش بینی میزان کارکنان.
- ۴- پیش بینی نیروی کاری نرم افزار.
- ۵- پیش بینی هزینه نرم افزار.
- ۶- پیش بینی جداول زمانی کار.

• به علت اختلافات تخمینهای ابزارهای خودکار لذا خروجی ابزارهای خودکار برآورد به عنوان امتیاز اطلاعاتی که از روی آنها تخمینها هدایت می شوند استفاده شوند و نه به عنوان منبع تخمین

فصل ۶: مدیریت و تحلیل ریسک

مقدمه

• عدم قطعیت باعث ریسک می شود

• **تحلیل و مدیریت مخاطرات چیست؟**

تحلیل ریسک (مخاطره) و مدیریت عبارت است از یک سری از اقدامات که یک تیم نرم افزاری کمک می کند تا عدم قطعیت (عدم اطمینان) را دریافته و آنرا مدیریت نمایند. مشکلات زیادی میتوانند برای یک پروژه نرم افزاری ایجاد اشکال نمایند.

ریسک یک مشکل بالقوه است ممکن است اتفاق بیفتد و ممکن است افعال نیفتد اما صرفنظر از پیامد آن، بهتر است که آن را شناسایی نماییم، احتمال وقوع آن را ارزیابی کنیم.

• **چه کسی این کار را انجام می دهد؟**

هر کسی که درگیر فرآیند نرم افزاری است در تحلیل و مدیریت مخاطرات مشارکت دارد، یعنی شامل :مدیران، مهندسين و مشتریان نرم افزار است

• **چه اقداماتی باید انجام داد؟**

اولین گام : **شناسایی ریسک** = آگاهی از اینکه چه اشکالاتی ممکن است پیش آید.

دومین گام : باید هر ریسکی **تحلیل** گردد تا احتمال وقوع آن و میزان خسارتی را که در صورت وقوع به بار می آورد تعیین شود. به محض اینکه این اطلاعات مشخص شدند، خطرات بر اساس احتمال وقوع و تاثیر طبقه بندی می شوند.

سومین گام : طرحی پیاده می شود تا خطراتی که احتمال وقوع آنها زیاد است و تاثیر آنها بیشتر است مدیریت نماید.

• **محصول کاری چیست؟**

طرحی برای متعادل سازی، کنترل و مدیریت ریسک. (RMMM) یا مجموعه صفحات حاوی اطلاعات ریسک

• **چگونه می توانیم اطمینان حاصل کنیم که این کار را درست انجام داده ایم؟**

ریسک هایی که تجزیه و تحلیل و کنترل می شوند باید حاصل مطالعه کامل افراد محصول فرآیند و پروژه باشند. **RMMM** با پیشرفت پروژه بایست بازبینی شود تا اطمینان حاصل شود ریسکها به روز نگه داشته می شوند. طرح های احتمالی برای کنترل و مدیریت ریسک باید واقعی باشند.

□ یک تعریف عقلانی و مفهومی از ریسک چیست؟

۱- ریسک به وقایع آینده ارتباط دارد.

امروز و دیروز در خارج از محدوده توجه قرار دارند، زیرا ما در زمان حال آنچه را که در گذشته با اعمال خود کاشته ایم درو می کنیم. سئوالی که مطرح می شود این است که پس آیا ما می توانیم با تغییر اعمال امروزمان فرصتی برای موقعیت متفاوت و شاید بهتر در آینده ایجاد نماییم.

۲- ریسک مستلزم تغییر است. همچون تغییر در ذهن، ایده، اعمال و یا مکان ها

(۳- در کتاب پرسمن) ریسک **مستلزم انتخاب** و عدم قطعیتی است که انتخاب ایجاب می نماید. بنابراین ریسک همچون مرگ و مالیات یکی از معدود امرهای مسلم زندگی است.

□ سه تعریف زیر بناتی ضمنی چارته از ریسک در رابطه با مهندسی نرم افزار:

۱. برای ما آینده اهمیت دارد :جلوگیری از خروج پروژه از مسیرش

۲. تغییر برای ما اهمیت دارد: چگونگی تاثیر تغییرات بر روی پروژه (تغییرات در نیازها- فناوری توسعه و ...)

۳. باید با انتخاب ها دست و پنجه نرم کنیم: انتخاب روشها و ابزارها، افراد، میزان تاکید بر افراد و....

□ نظریه پتر دراگر :

همانگونه که تلاش برای حذف خطری ثمر است، و تلاش برای کاستن آن سوال برانگیز می باشد، ضروری است خطراتی را که برای آنها وقت صرف می کنیم خطرات بجایی باشند. قبل از اینکه خطرات بجایی را که در طی یک پروژه نرم افزاری به آنها خواهیم پرداخت **شناسایی** نماییم، حائز اهمیت است که کلیه خطراتی را که هم برای مدیران و هم برای کارورزان مشهود است شناسایی نماییم.

راهبرد واکنش بر ریسک

راهبردهای ریسک واکنشی به نام "مدرسه مدیریت ریسک Indiana Jones" نامیده می شوند. در فیلمهای سینمایی که ایندیانای جونز وجود دارد او با خطرات عمده ای روبروست و همیشه می گوید : "نگران نباشید راهی پیدا خواهم کرد!"

یک مدیر پروژه متوسط در برابر مشکلات ، ایندیانای جونز نیست و اعضای تیم مورد اعتمادش نمی باشند

اکثریت تیم های نرم افزاری منحصراً بر روی راهبردهای ریسک واکنش پذیر تکیه دارند.

مهمتر از همه اینکه یک راهبرد واکنش پذیر پروژه را برای خطرات احتمالی تحت **کنترل (نظارت)** دارد. منابعی برای مقابله با آنها در نظر گرفته می شود. تا زمانی که تبدیل به **مشکل واقعی** شدند از آنها استفاده شود. معمولاً تیم نرم افزاری در مورد ریسک کاری انجام نمی دهند تا اینکه مشکلی پیش بیاید. در این حال تیم وارد عمل شده و **سریعاً برای حل مشکل** اقدام می نماید. این وضعیت غالباً «**وضعیت پرواز آتشی (درست: جنگ با آتش)**» نامیده می شود. وقتی که این کار با **شکست** مواجه می شود «**مدیریت بحران**» کنترل را بدست می گیرد که در این حال پروژه با خطر واقعی مواجه می گردد.

☺ fire fighting **Flight**

راهبرد واکنشهای پیش کنش (غیر واکنشی) در ریسک

راهبردهوشمندانه پیش کنش خیلی پیش از آنکه کار فنی آغاز گردد شروع می شود. خطرات بالقوه (دارای شانس وقوع) **شناسایی** شده، احتمال و تاثیر آنها مورد **ارزیابی** قرار گرفته و سپس بر اساس اهمیت **طبقه بندی** می شوند. سپس تیم نرم افزاری **طرحی** را برای کنترل ریسک ارائه می دهد. هدف اولیه **جلوگیری از بروز ریسک** (خطر) است اما از آنجایی که تمام خطرات اجتناب ناپذیر هستند، تیم یک طرح احتمالی ارائه می دهد تا بتواند به یک روش کنترل شده و موثر واکنش نشان دهد.

ریسک های نرم افزار

در تعریفهای ریسک همواره دارای دو **مشخصه مشابه** وجود دارد:

• **عدم قطعیت**: رویدادی که مشخص می کند آیا ریسکی به وقوع خواهد پیوست یا خیر؛ یعنی هیچ ریسکی که احتمال وقوع آن ۱۰۰٪ باشد وجود ندارد.

• **خسارت** : اگر ریسک یک واقعیت باشد، پیامدها و یا خسارات ناخواسته ای بوقوع خواهد پیوست

زمان تحلیل ریسک ، تعیین **عدم اطمینان (قطعیت)** به صورت کمی و درجه فقدان ناشی از هر ریسک اهمیت زیادی دارد.



□ دسته های مختلف ریسک ها

□ ریسک پروژه:

بر برنامه ریزی پروژه تاثیر دارند. اگر ریسکهای پروژه به واقعیت بپیوندند این احتمال وجود دارد که برنامه پروژه با اشتباه مواجه شود و قیمتها افزایش یابند. ریسک های پروژه، بودجه، برنامه زمان بندی، پرسنل، منابع، مشتری بالقوه و مشکلات مربوط به نیازمندی ها و تاثیر آنها را بروی پروژه نرم افزاری شناسایی می کنند

□ خطرات فنی (ریسکهای تکنیکی) :

وقتی که خطر فنی به واقعیت می پیوندد، پیاده سازی دشوار و یا غیر ممکن می گردد. خطرات فنی مشکلات بالقوه مربوط به طراحی، پیاده سازی، رابط، تعین صحت و نگهداری را شناسایی می کنند. بعلاوه ابهام مربوط به ویژگی، عدم قطعیت فنی، زوال فنی و فناوری «پیشرو» از عوامل و فاکتور مربوط به ریسک هستند. خطرات فنی بدین جهت بوقوع می پیوندند که حل مشکل از آنچه که ما تصور می کنیم دشوارتر می باشد.

□ خطرات تجاری

عوامل ریسکهای تجاری:

• مثال : ساختن فیلم

۱. ساخت یک محصول و یا سیستم بسیار عالی که هیچکس واقعاً طالب آن نیست. (ریسک بازار)
۲. ساخت محصول که دیگر مناسب راهبرد تجاری کلی برای شرکت مزبور نمی باشد. (ریسک راهبردی)
۳. ساخت محصولی که نیروهای فروش نمی دانند چگونه آن را بفروشند (ریسک فروش)
۴. از دست دادن حمایت مدیریت ارشد بدلیل تغییر نقطه عطف (دیدگاه) و یا تغییر افراد (ریسک مدیریتی)
۵. از دست دادن بودجه و یا تعهد پرسنل. (ریسک بودجه)

□ ریسکهای قابل پیش بینی (شناخته شده)

آنهايي هستند که مي توانند بعد از ارزیابی دقیق پروژه، خطرات فنی و خطرات تجاری و آشکار شدن منابع قابل اعتماد (مثل ثبات تاریخ تحویل یا عدم ناقص بودن مستندات) مشخص شوند. ریسکهای قابل پیش بینی هستند که از تجربه های پروژه های قبلی استخراج می شوند (مثلا ارتباط ضعیف با مشتری)

□ ریسکهای غیر قابل شناسایی

ریسکهایی که به وقوع می پیوندند و شناسایی آنها از قبل بسیار دشوار است (مثل جوکر در بازی ورق)

شناسایی ریسک

شناسایی ریسک عبارت است از یک تلاش نظام مند برای تعیین تهدیدهایی که متوجه طرح پروژه هستند (برآوردها، برنامه ها، تخصیص منابع و غیره) و با شناسایی ریسکهای قابل پیش بینی مدیر پروژه اولین گام را در راه اجتناب از آنها و در صورت امکان کنترل آنها به کار می برد

انواع کلی ریسک در مرحله شناسایی

۱. ریسک های محصول ویژه (خاص محصول) : را تنها می توان توسط کسانی تشخیص داده شوند که درک روشنی از فناوری، مردم و محیطی که مختص پروژه دارند

۲. ریسک عمومی : ریسک های عمومی تهدید بالقوه ای برای هر نوع پروژه نرم افزاری می باشند.
برای شناسایی ریسک های محصول ویژه (خاص محصول)، طرح پروژه و گزارش نرم افزاری از حوزه مورد (گزارش دامنه نرم افزار) بررسی قرار گرفته و پاسخی برای سوال مقابل داده می شود. "چه ویژگی های خاصی از این محصول می تواند طرح پروژه ما را به خطر اندازد؟"

یک روش کاربردی برای شناسایی ریسک ها عبارت است از تهیه یک چک لیست حاوی موارد ریسک از چک لیست می توان برای شناسایی ریسک و تمرکز (نقطه عطف) به مسائل فرعی شناخته شده و قابل پیش بینی ریسک استفاده کرد.

طبقه های ریسک

- اندازه محصول (PS): ریسک هایی که مرتبط به اندازه کلی نرم افزار شناخته شده و یا تغییر یافته هستند.
- تاثیر تجارت (BU): ریسک هایی که مربوط به محدودیت های ناشی از مدیریت و یا بازار تجاری است.
- ویژگی های مشتری (CM): ریسک های مربوط به پیچیدگی مشتری و توانایی سازندگان برای ایجاد ارتباط زمانبندی شده با او
- تدوین فرآیند (PD): ریسک های مربوط به میزان تعریف فرآیند نرم افزار که توسط سازمان توسعه دهنده دنبال می شود.
- محیط توسعه (EV): ریسک های مربوط به در دسترس بودن و کیفیت ابزارهایی که برای ساخت محصول مورد استفاده است.
- فناوری ساخت (TE): ریسک های مربوط به پیچیدگی سیستم ساخته شده و «تازگی» فناوری که سیستم با آن ساخته می شود.
- اندازه و تجربه کارکنان (ST): ریسک های مربوط به تجربه پروژه فنی کلی مهندسان نرم افزاری که کار را انجام می دهند.

ارزیابی(؟)(شناسایی) ریسک کلی پروژه:

سوالات زیر از گروهی داده های ریسک نتیجه گیری شده اند که توسط مدیران با تجربه در نقاط مختلف دنیا بدست آمده اند و برحسب اهمیت مرتب شده اند:

۱. آیا مدیران نرم افزار و مشتری رسماً اقدام به حمایت از پروژه نموده اند؟

۲. آیا مصرف کنندگان نهایی در خصوص پروژه و سیستم / محصول ساخته شده اظهار نظر کرده اند؟

۳. آیا تیم مهندسی نرم افزار و مشتریان آنها نیازمندیها را کاملاً درک می نمایند؟

۴. آیا مشتریان کاملاً در تعریف نیازمندیها نقش دارند؟

۵. آیا مصرف کنندگان نهایی انتظارات واقع بینانه دارند؟

۶. آیا حوزه پروژه ثابت(دامنه پایدار) است ؟

۷. آیا تیم مهندسی نرم افزار ترکیب درستی از مهارت ها دارند؟

۸. آیا نیازمندیها پروژه ثابت است؟

۹. آیا تیم پروژه تجربه ای در خصوص فناوری در حال انجام دارند ؟

۱۰. آیا تعداد افراد تیم پروژه برای انجام کار کافی هستند؟

۱۱. آیا کلیه مشتریان/مصرف کنندگان در خصوص اهمیت پروژه و در خصوص شرایط سیستم/ محصول ساخته شده با هم توافق دارند؟

در صورت منفی بودن جواب همه سؤالاها پروژه به صورت کلی همه مراحل نظارت **بدون خطا** انجام می شود

درجه ریسک پروژه به طور مستقیم **متناسب** است با **تعداد پاسخهای منفی** به این سؤالات

اجزاء و محرک های (هدایت کننده) ریسک:

بر طبق رساله نیروی هوایی آمریکا در جهت شناسایی و کاهش ریسکهای نرم افزاری است. در این روش مدیر پروژه می بایست **محرکهای ریسک** که روی **اجزای ریسک** نرم افزاری را **تاثیر** می گذارند را شناسایی کند. **تاثیر** محرکهای ریسک می تواند به چهار دسته **جزئی، مرزی، بحرانی یا فاجعه آمیز** باشد. در این رساله **اجزای ریسک** عبارتند از: (کارایی-هزینه-حمایت و زمانبندی)

زمانبندی	هزینه	پشتیبانی	عملکرد	مؤلفه ها
				فاجعه
1	Failure results in increased costs and scheduled delays with expected values in excess of \$500k	Failure to meet the requirement would result in mission failure		
2	Unachievable LOC	Significant degradation To nonachievement of technical performance	Nonresponsive or unsupportable software	
				بحر
1	Failure results in operational delays and/or increased costs with expected value of \$1 to \$100 k	Failure to meet the requirement would degrade system performance to a point where mission success is questionable		
2	Possible slippage in LOC	Some reduction in technical performance	Minor delay delays in software modifications	
				مرز
1	Costs, impacts, and/or recoverable schedule slips with expected value of \$1 to \$100 k	Failure to meet the requirement would result in degradation of secondary mission		
2	Realistic, achievable schedule	Minimal to small reduction in technical performance	Responsive software support	
				قابل
1	Error results in minor cost and/or schedule impact with expected value of less than \$1 k	Failure to meet the requirement would create inconvenience or nonoperational impact		
2	Early achievable LOC	No reduction in technical performance	Easily supportable software	

۱. **ریسک عملکرد**: مدیران عدم قطعیتی که به کمک آن نیازهای پروژه به هدف مورد نظر منطبق می شوند.

۲. **ریسک هزینه**: میزان عدم قطعیتی که به کمک آن بودجه پروژه تامین می گردد.

۳. **ریسک پشتیبان**: میزان عدم قطعیتی که اصلاح، تطابق و تقویت نرم افزار، آسان می شود.

۴. **ریسک برنامه**: میزان عدم قطعیتی که زمانبندی پروژه رعایت شود و محصول بموقع ارائه می گردد.

اگر میزان عدم قطعیت ۰ باشد (قطعی) یعنی ریسک و خطری وجود ندارد

• **تاثیر محرکهای ریسک** که روی **اجزای ریسک**:

۱. قابل صرف نظر (اغماض) - جزئی

۲. مرزی

۳. بحرانی

۴. فاجعه آمیز

جدول ارزیابی تاثیرات

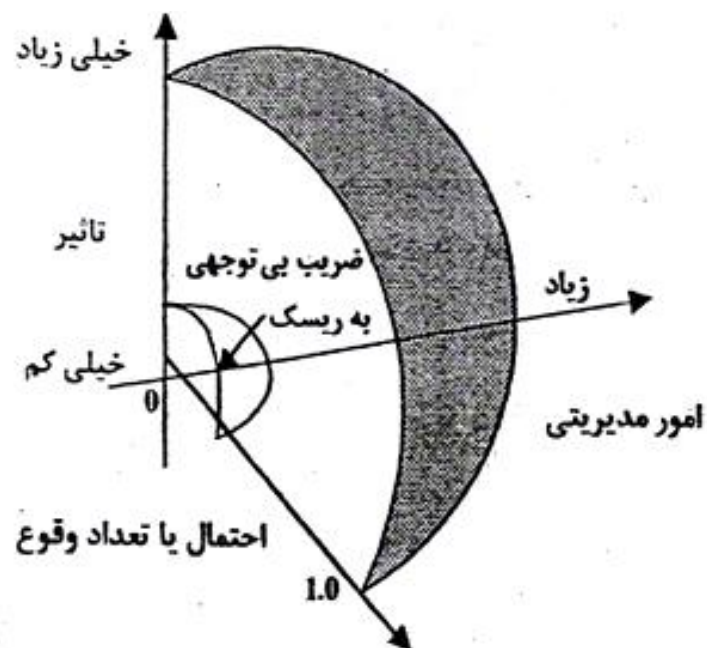
تصویر (تخمین) ریسک

- برآورد ریسک ، که تخمین ریسک نیز نامیده می شود. سعی در تعیین مقدار ریسک به دو روش دارد :
- ۱- احتمال و یا امکان اینکه ریسک و مخاطره واقعیت یابد و روی دهد. ۲- پیامدها و مشکلاتی که در صورت روی دادن آن ریسک برنامه ریز پروژه همراه با مدیران دیگر برای تصویر (برآورد) ریسک چهار فعالیت انجام می دهند
 ۱. ایجاد مقیاسی که منعکس کننده احتمال مشاهده شده یک ریسک است.
 ۲. مشخص کردن پیامدهای ریسک.
 ۳. برآورد تاثیر ریسک بر روی پروژه و محصول
 ۴. توجه به صحت و دقت کلی برآورد ریسک بطوری که هیچ درک نادرستی وجود نداشته باشد.

ساخت و توسعه یک جدول ریسک:

مخاطرات (ریسکها)	طبقه	احتمال	تاثیر	م.ن.ت. ریسک
Size estimate may be significantly low	PS	60%	2	
Larger number of users than planned	PS	30%	3	
Less reuse than planned	PS	70%	2	
End users resist system	BU	40%	3	
Delivery deadline will be tightened	BU	50%	2	
Funding will be lost	CU	40%	1	
Customer wil change requirements	PS	80%	2	
Technology will not meet expectations	TE	30%	1	
Lack of training on tools	DE	80%	3	
Staff inexperienced	ST	30%	2	
Staff inexperienced	ST	60%	2	
Staff turnover will be high				

- یک تیم پروژه با فهرست بندی تمام ریسک ها (صرفنظر از فاصله زمانی وقوع آن) در اولین ستون جدول آغاز می گردد.
 - هر ریسک در ستون دوم طبقه بندی شده است.
 - احتمال وقوع هر ریسک در ستون بعدی است. میزان احتمال هر ریسک را اعضای تیم تخمین بزنند. (بررسی محرکهای ریسک غیرممکن/نامحتمل/محتمل/احتمال زیاد)
 - میانگین مقوله های هر چهار جزء ریسک (اجزا ریسک: عملکرد، پشتیبانی ، هزینه و برنامه زمانی) برای تعیین ارزش کلی تاثیر برآورده می گردد.
 - با تکمیل ۴ ستون اول ، جدول بر اساس ستون احتمال و تاثیر مرتب می شود
- مقادیر تاثیر : ۱- فاجعه ۲- مرزی ۳- بحرانی ۴- قابل اغماض
م.ن.ت. مدیریت - نظارت - تخفیف (RMMM)
CU مشتری و PS سباز پروژه TE کارکنان BU تجاری DE محیط توسعه .



• مدیر پروژه جدول مرتب شده حاصل را مورد بررسی قرار داده و یک خط فرضی (برشی) تعریف می کند «خط فرضی» بیانگر این است که فقط به ریسک هایی که در بالای خط قرار گرفته اند توجه بیشتری می شود. ریسک هایی که در پایین خط قرار می گیرند مجدداً مورد بررسی قرار می گیرند تا اولویت بندی مرحله دوم انجام پذیرد

• تمام ریسک هایی که در بالای خط فرضی قرار می گیرند باید کنترل گردند. ستونی که تحت عنوان م.ن.ت ریسک (RMMM) (مدیریت - نظارت - تخفیف) نام گرفته دارای نشانگری است به طرف ۱ تخفیف و تعدیل ، ۲ کنترل و نظارت و ۳ مدیریت ریسک و یا عبارت دیگر مجموعه ای از چند صفحه اطلاعات در خصوص ریسک که برای کلیه ریسک های که در بالای خط فرضی قرار گرفته اند در نظر گرفته شده اند.

ارزیابی میزان اثر ریسک

• بررسی نتایج وقوع ریسک تأثیر می گذارند: محرز عوامل موثر بر نتایج ریسک؟

۱. ماهیت ریسک : بیانگر مشکلاتی است که در صورت وقوع ریسک بروز خواهند نمود.

۲. حوزه و محدوده ریسک : سختی آن را (میزان جدی بودن) با پراکندگی و توزیع کلی (چه مقدار از پروژه و چند مشتری دچار مشکل می شوند) ترکیب می نماید

۳. زمان ریسک: بیانگر این است که چه وقت و برای چه مدتی تأثیر آن محسوس خواهد بود.

در بیشتر موارد یک مدیر پروژه خواهان این است که «خبر های ناخوشایند» هرچه زودتر برسد . اما در برخی مواقع ترجیح می دهند که این خبرها دیرتر برسند.

❖ مراحل تعیین نتایج کلی ریسک (از دید نیروی هوایی آمریکا)

۱. برای هر یک از اجزا ریسک مقدار احتمال متوسط وقوع را تعیین کنید
۲. میزان تاثیر هر جز را با استفاده از معیار (جزئی، مرزی، بحرانی یا فاجعه آمیز) تعیین کنید
۳. تکمیل جدول ریسک و تحلیل نتایج

❖ **میزان در معرض ریسک قرار گرفتن** کلی را **RE** نامیده می شود و می توان با استفاده از رابطه زیر تعیین کرد.

$$RE = P * C$$

\swarrow \searrow
 احتمال وقوع یک ریسک هزینه وارده به پروژه در صورت وقوع ریسک

❖ مثال

• **تشخیص ریسک** : ۷۰٪ از اجزا برای استفاده مجدد برنامه ریزی شده اند

• احتمال ریسک : ۸۰٪

هزینه هر LOC : 14\$

اندازه هر جزء : 100 loc

• تعداد اجرا : ۶۰ جزء

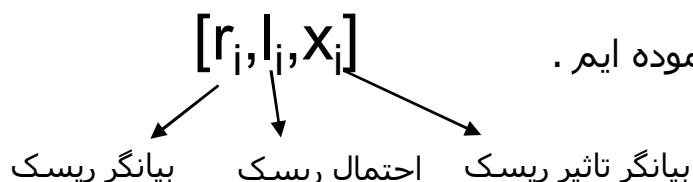
حل: تعداد اجزایی که در اثر ریسک دچار مشکل می شوند : $60 * 30\% = 18$

هزینه اجزای دچار مشکل شده : $18 * 100 * 14 = 25200\$$

میزان در معرض ریسک بودن $RE = 0.8 * 25200 = 20200\$$

تشخیص – (ارزیابی) ریسک

در این نقطه از فرآیند مدیریت ریسک، ما یک مجموعه سه تایی به شکل زیر ایجاد نموده ایم.



• سطوح ارجاعی (سطح مبنایی برای تشخیص ریسک):

برای اینکه ارزیابی سودمند واقع شود باید یک سطح ارجاعی برای ریسک تعریف نمود. در مورد بیشتر پروژه های نرم افزاری، اجزاء ریسک، (یعنی عملکرد، هزینه، پشتیبانی و زمان بندی) نیز بیانگر سطوح ارجاعی ریسک می باشند. یعنی سطحی برای کاهش کارایی و عملکرد، بالا رفتن هزینه، اشکال در پشتیبانی و یا کم و کاستی در برنامه و زمانبندی که باعث اتمام پروژه می شوند وجود دارد. اگر ترکیبی از ریسکها سبب ایجاد مشکلاتی شوند که منجر به **ار حد گذشتن این سطوح ارجاعی** گردد، **کار متوقف** خواهد شد.

سطوح ارجاعی ریسک دارای **نقطه منفردی** است که **نقطه ارجاعی و یا نقطه شکست** نامیده می شود. که در این نقطه تصمیم گیری برای ادامه پروژه و یا ختم آن (مشکلاتی که خیلی بزرگ هستند) اهمیت یکسانی دارد.

❖ اقدامات زمان ارزیابی ریسک :

۱. **سطوح ارجاعی ریسک** را برای پروژه تعیین نمایید.

۲. سعی کنید بین هر (r_i, I_i, X_i) و هر یک از سطوح ارجاعی یک رابطه ایجاد کنید.

۳. مجموعه از **نقاط ارجاعی** را پیش بینی کنید و تعیین کننده یک ناحیه پایان است و بوسیله یک منحنی و یا نواحی عدم قطعیت محدود می شود.

۴. سعی کنید پیش بینی نمائید (تاثیر) **ترکیب های اجزاء ریسک ها** چگونه سطح ارجاعی را تحت تاثیر قرار خواهند داد.

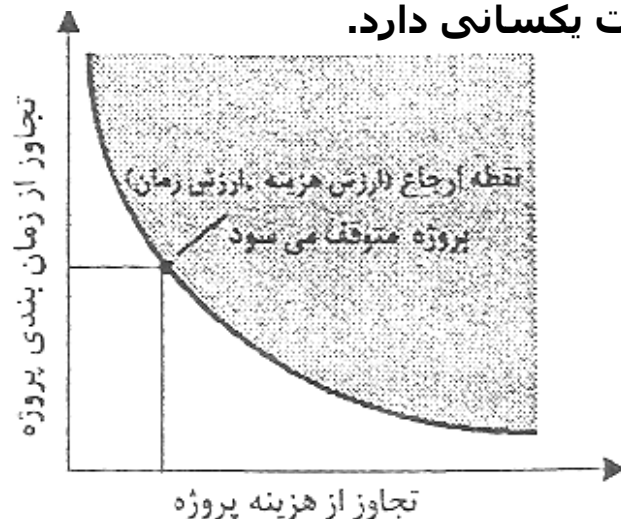
پالایش ریسک

با گذشت زمان از آغاز پروژه نکات بیشتری در مورد پروژه و ریسک بدست می آید و ممکن است یک ریسک به مجموعه از ریسکها یا جزئیات بیشتر پالایش شود که هر یک برای تخفیف (کاهش)، نظارت و مدیریت ساده تر می باشند. برای این منظور می توان ریسک را در قالب CTC (وضعیت-انتقال-نتیجه) نمایش داد

با شرایط اینکه <وضعیت> آنگاه باید توجه داشت که (احتمالاً) <نتیجه>

شرط

condition-transition-consequence=CTC





• نحوه تدوین وظیفه عمومی در قالب CTC

- وضعیت فرعی ۱: اجزاء ویژه قابل استفاده مجدد توسط شخص ثالثی که آگاهی از استاندارد های طراحی ندارد ساخته شده اند.
 - وضعیت فرعی ۲: طرح استاندارد برای سطوح مشترک اجزاء هنوز به حالت یکپارچه در نیامده است و ممکن است با اجزاء خاص قابل استفاده مجدد موجود سازگاری پیدا نکند.
 - وضعیت فرعی ۳: اجزاء خاص قابل استفاده در **زبانی** بکار گرفته شده است که در محیط هدف مورد پشتیبانی قرار نگرفته است.
- پالایش به جدا سازی ریسکهای مرتبط کمک می کند و ممکن است باعث ساده تر شدن تحلیل شود

تخفیف (کاهش)، نظارت و مدیریت ریسک:

هدف تجزیه و تحلیل ریسک **ارائه راهبردی** برای پرداختن (مقابله) به ریسک است
یک راهبرد موثر و کارا باید سه مسئله را در نظر بگیرد:

❖ **اجتناب از ریسک**

❖ **کنترل ریسک**

❖ **مدیریت ریسک و برنامه ریزی احتیاطی**

چنانچه تیم نرم افزاری یک روش **پیش کنش** برای ریسک انتخاب کند، **اجتناب** بهترین راهبرد خواهد بود. این کار از طریق ارائه طرح برای تعدیل و تخفیف ریسک امکان پذیر است. و بایست برای کاهش عواملی که ریسک بالا محسوب می شوند اقدامات پیشگیرانه صورت گیرد

مدیر پروژه فاکتورهایی را **کنترل** و نظارت می کند که می توانند مشخص نمایند **هر ریسک با چه احتمالی** اتفاق می افتد و نیز بر موثر بودن مراحل کاهش ریسک نظارت می کند.

مدیریت ریسک و برنامه ریزی احتیاطی فرض می کند که فعالیتهای کاهش ریسک شکست خورده اند و **ریسک اتفاق افتاده** لذا مدیر پروژه بایست برنامه ریزی لازم از جمله وجود نیروی پشتیبان را در نظر گرفته باشد. که گاهی هزینه ها را بالا می برد و شاید با افزایش میزان ریسکها، مدیریت ریسکها وقت گیر هم باشد. لذا طبق **قانون ۲۰-۸۰**، با بررسی ۲۰ درصد ریسکها ۸۰ درصد ریسکها حل می شوند. و وقت کمتری برای مدیریت مورد نیاز است

مخاطرات و ریسک های ایمنی:

ریسک تنها به پروژه نرم افزاری محدود نمی شود. ریسک ها ممکن است پس از اینکه پروژه با موفقیت انجام شده و تحویل مشتری گردید، بروز کنند. این ریسک ها غالباً مربوط به پیامدهای عدم موفقیت نرم افزار در زمان اجرا است. خطاها معمولاً حاصل از خطاهای انسانی اند

خطاهایی که در کنترل قراردادی سخت افزار می توان نادیده گرفت و بر آنها سرپوش گذاشت، بهنگام استفاده از نرم افزار نادیده گرفتن آنها بسیار دشوارتر خواهد بود.

ایمنی نرم افزار و تحلیل خطر : عبارتند از فعالیت های مربوط به اطمینان از کیفیت نرم افزار که به شناسایی و ارزیابی خطرات بالقوه ای می پردازد که می تواند نرم افزار را بطور منفی تحت تاثیر قرار داده و سبب بروز نقص در یک سیستم کامل گردد

طرح RMMM

• یک راهبرد مدیریت ریسک را می توان در طرح پروژه نرم افزاری گنجاند و یا مراحل مدیریت ریسک را می توان بصورت یک طرح تعدیل ، کنترل و یا مدیریت سازماندهی نمود. طرح RMMM از تمام کارهایی که به عنوان بخشی از تحلیل ریسک انجام می پذیرد و توسط مدیر پروژه بعنوان بخشی از طرح پروژه کلی بکار می رود، گزارش مستند تهیه می نماید.

• برخی تیمهای نرم افزاری به جای قالب RMMM ، هر ریسک را به صورت مجزا با استفاده از صفحه اطلاعات ریسک (RIS) مستند می کنند، که این کار معمولاً توسط سیستم بانک اطلاعاتی انجام می شود

• با شروع پروژه و مستند شدن RMMM ، کاهش ریسک (اجتناب از ریسک) و مراحل نظارت (دنبال نمودن پروژه) شروع می گردد

قالب RIS در شکل زیر نمایش داده شده است.

Risk information sheet			
Risk ID: P02-4-32	Date: 5/9/02	Prob: 80%	Impact: high
Description: Only 70 percent of the software components scheduled for reuse will, in fact, be integrated into the application. The remaining functionality will have to be custom developed.			
Refinement/context: Subcondition 1: Certain reusable components were developed by a third party with no knowledge of internal design standards. Subcondition 2: The design standard for component interfaces has not been solidified and may not conform to certain existing reusable components. Subcondition 3: Certain reusable components have been implemented in a language that is not supported on the target environment.			
Mitigation/monitoring: 1. Contact third party to determine conformance with design standards. 2. Press for interface standards completion; consider component structure when deciding on interface protocol. 3. Check to determine number of components in subcondition 3 category; check to determine if language support can be acquired.			
Management/contingency plan/trigger: RE computed to be \$20,200. Allocate this amount within project contingency cost. Develop revised schedule assuming that 18 additional components will have to be custom built; allocate staff accordingly. Trigger: Mitigation steps unproductive as of 7/1/02			
Current status: 5/12/02: Mitigation steps initiated.			
Originator: D. Gagne		Assigned: B. Laster	

فصل ۷: زمانبندی و ردگیری پروژه

داستان مهندس جوان

در اواخر دهه ۱۹۶۰، یک مهندس جوان برای نوشتن یک برنامه کامپیوتری برای یک کاربرد ساخت اتوماتیک انتخاب شد. دلیل انتخاب او ساده بود. او تنها فردی در گروه تکنیکی خود بود که در یک سمینار برنامه نویسی کامپیوتری شرکت کرده بود. او از ورودی‌ها و خروجی‌های زبان اسمبلی و FORTRAN آگاه بود ولی چیزی در مورد مهندسی نرم افزار و حتی کمی در مورد زمانبندی و پیگیری پروژه نمی دانست.

رئیس او راهنماهای مناسبی را در اختیارش گذاشت و برای او کاری را که باید انجام شود توضیح داد. به او گفته شد که پروژه باید در two ماه تکمیل شود.

او راهنماها را مطالعه نمود، روش را در نظر گرفت، و کدنویسی را بعد از دو هفته شروع کرد، رئیس او را به دفتر خود فراخواند در مورد نحوه پیشرفت کار از او سؤال نمود.

"بسیار عالی"، پاسخی بود که مهندس جوان با غرور و اطمینان به رئیس ارائه نمود. "این پروژه ساده تر از آن چیزی بود که فکر می کردم. احتمالاً نزدیک ۷۵ درصد کار تمام شده است".

رئیس لبخندی زد و گفت "بسیار عالی است"، و مهندس جوان را تشویق به انجام کار نمود. آنها برای هفته آینده برنامه ملاقات را تنظیم نمودند.

هفته بعد رئیس، مهندس را به دفترش فراخواند و پرسید، "در کجا هستیم".

مهندس جوان پاسخ داد "همه چیز خوب پیش می رود اما چند مشکل کوچک وجود دارد. آنها را

برطرف خواهم نمود و به مسیر اصلی باز خواهم گشت".

رئیس پرسید: "در مورد مهلت چه نظری دارید؟"

مهندس جوان گفت: "مشکلی نیست، نزدیک به ۹۰ درصد کار تکمیل شده است".

اگر شما در دنیای نرم افزار برای بیش از چند سال کار کرده باشید، پایان داستان را می دانید. تعجبی نخواهد داشت که مهندس جوان در کل زمان پروژه فقط ۹۰ درصد آن را تکمیل نمود (با کمک دیگران) و فقط یک ماه دیرتر کار خود را به پایان رساند.

این داستان دهها هزار بار در بین توسعه دهندگان نرم افزار در سه دهه گذشته اتفاق افتاده است. یک سؤال مهم این است که چرا؟

□ زمان بندی و ردگیری پروژه چیست؟

شما يك مدل فرآیند مناسب انتخاب کرده اید، و کارهایی که مهندس نرم افزار باید انجام دهد را مشخص کرده اید، و برآورد مقدار کار و تعداد افراد را انجام داده اید و ریسکها را ارزیابی کرده اید.

زمانبندی و پیگیری طرح عبارت است از وصل کردن **نقطه ها به هم دیگر (نقطه بازی؟؟؟؟)**، شبکه از وظائف مهندسی نرم افزار را به وجود می آورد که باعث انجام به موقع کارها می شود. بعد از ایجاد شبکه هم بایست مسئولیت برای هر کار اختصاص یابد، اطمینان از انجام کار بدست آید و در صورت عملی شدن ریسکها بایست با این شبکه منطبق شود

□ چه کسی آنرا انجام می دهد؟

در سطح پروژه مدیر پروژه، از طریق اطلاعات حاصل از مهندسين نرم افزار، زمانبندی و ردگیری را انجام می دهد و در سطح انفرادي مهندسين نرم افزار خود انجام می دهند

□ دلیل اهمیت آن چیست؟

در سیستمهای پیچیده به دلیل تاثیر عمقی نتیجه کاری يك مرحله روی سایر مراحل و امکان انجام فعالیتهای موازی، درک ارتباطات متقابل بدون داشتن يك برنامه زمانبندی بسیار مشکل است

همچنین ارزیابی پیشرفت برای پروژه های بزرگ یا متوسط بدون يك برنامه زمانبندی جزئی غیر ممکن است

□ مراحل انجام آن کدامند

پالایش وظایف مهندسی نرم افزار، و انتساب نیروی کار و زمان لازم به هر وظیفه و ایجاد شبکه وظایف برای تحویل به موقع نرم افزار

□ چگونگی اطمینان از صحت انجام کار زمانبندی . ردگیری؟

زمانبندی مناسب به موارد زیر نیاز دارد:

۱- ظهور تمام وظایف در شبکه

۲- اختصاص زمان و تلاش آگاهانه به هر وظیفه

۳- نشان دادن مناسب ارتباط متقابل بین وظایف ۴- اختصاص منابع به وظایف و کارها

۵- در نظر گرفتن شمارنده های نزدیک به هم به طوریکه پیشرفت کار قابل پیگیری باشد

مفاهیم اولیه

عوامل اصلی که باعث تاخیر در تحویل نرم افزار می شوند :

- یک مهلت غیر قابل تحقق توسط فردی خارج از گروه توسعه نرم افزار ایجاد شده و فشاری بر مدیران و مجریان وارد نموده است.
 - تغییر نیازهای مشتری که در تغییرات زمانبندی پیش بینی نشده اند.
 - کوچک شمردن تخمین میزان فعالیت و تعداد منابعی که برای انجام کار مورد نیاز است.
 - ریسک های قابل پیش بینی و غیر قابل پیش بینی که در زمان شروع پروژه در نظر گرفته نشده اند.
 - مشکلات تکنیکی که از قبل قابل پیش بینی نبودند.
 - مشکلات انسانی که از قبل قابل پیش بینی نبودند.
 - ارتباط نادرست بین سرپرست های پروژه که باعث تأخیر می گردد.
 - شکست مدیریت پروژه برای تشخیص این که پروژه فراتر از زمانبندی پیش خواهد رفت، به همراه فقدان انجام عملی برای اصلاح این مشکل.
- موعدهای غیر قابل تحقق (بی حساب کتاب) ، حقایق از زندگی در تجارت نرم افزار می باشند . بعضی وقتها از نظر فردی که موعد را مقرر کرده است مشروع است ولی عقل سلیم می گوید که مشروعیت را باید کسانی که کار را انجام می دهند تعیین کنند

توضیحاتی بر دیر کرد

ناپلئون گفته است: "هر فرمانده که مسئول اجرای طرح می باشد، و فکر می کند با شکست روبرو می شود در اشتباه است. باید بر دلایل خود تأکید گذارد، بر تغییر طرح اصرار داشته باشد، و در نهایت استعفا دهد، به جای این که ابرازی باشد برای شکست ارتش خود".

اگر بهترین ارزیابی ها نشان می دهند که موعد مقرر ، غیر قابل تحقق است يك مدیر پروژه خوب باید ، گروه خود را در برابر فشارها (زمانبندی) حمایت کند و این فشارها را به منشاءشان باز گرداند

□ يك مثال

پروژه اي براي تحويل ۹ ماهه درخواست شده كه با انجام برآوردها مشخص مي شود حداقل ۱۴ ماه نياز است
تغيير زمان تحويل ميسر نيست از سويي رد كردن پروژه هم بسيار مضر است در اين زمان انجام اقدامات زير توصيه مي شود:

۱. مهلت مورد نياز واقعي را با استفاده از داده هاي تاريخي و تجربه بصورت مشروح مشخص كنيد
 ۲. با استفاده از مدل فرايند افزايشي ، يك راهبرد مهندسي توليد كنيد كه عملكردهاي مهم (اصلي) را در مهلت خواسته شده انجام دهد و وظايف ديگر را به آينده موكل مي كنيم
 ۳. با مشترى ملاقات كنيد و با استفاده از برآوردها پروژه هاي قبلي توضيح دهيد كه مهلت خواسته شده ؛ غير واقع بينانه است و با اطمينان وقت لازم براي تحقق كامل پروژه را طلب كنيد
 ۴. به عنوان راه چاره مدل افزايشي را به عنوان جايگزيني ارائه دهيد (ارائه راه حلهاي ديگر:)
- مي توان افزايش بودجه را با افزايش تعداد نيروي كار و احتمالا پايين آمدن كيفيت پيشنهاد كرد
انجام ندادن برخي از قابليت هاي و كارايي هاي نرم افزار براي تحويل با كيفيت در ۹ ماه يا همه كارايي ها براي تحويل ۱۴ ماهه بي خيال شدن و اميدواري براي اتمام ۹ ماهه كه ممكن است به از دست دادن مشتري منجر شود.

اصول پایه

لزوم ردگیری فعالیتهای بحرانی

«يك وقتي در يك روزي»

از Fred Brooks كه مؤلف شناخته شده ای برای كتاب Mythical Man – Month می باشد، سؤال شده است كه چگونه پروژه های نرم افزاری خارج از زمانبندی خود قرار می گیرند. پاسخ او عمیق و ساده بود:

يك پروژه نرم افزاري شامل صدها كار كوچك است كه بايد براي دستيابي به هدف بزرگتر انجام شوند. برخي كارها بدون نگراني از تاثير روي كل پروژه تكميل مي شوند ولي بعضي كارهاي ديگر روي "مسير بحراني" قرار مي گيرند كه اگر اين كارهاي بحراني به تعويق بيفتند ، تاريخ اتمام كل پروژه به اتمام مي افتد (پيش نيازند مثل دروس پيش نياز دانشگاه)



□ هدف مدیر پروژه :

۱- توصیف همه کارهای پروژه ۲- ساختن شبکه از روابط متقابل ۳- شناسایی کارهای بحرانی

۴- پیگیری پیشرفت آنها برای اطمینان از جلوگیری "یک وقتی در یک روزی" (نیازمند ارائه برنامه ریزی)

□ تعریف زمان بندی پروژه نرم افزاری :

فعالیتی است که نیروی انسانی برآورد شده را در مدت برنامه ریزی شده پروژه، با تخصیص فعالیت به کارهای مهندسی نرم افزار توزیع می نماید

برنامه زمانبندی با گذشت زمان کامل می شود.

□ فازهای زمانبندی برای پروژه نرم افزاری :

۱- زمانبندی ماکروسکوپی : تمام فعالیت‌های عمده مهندسی نرم افزار شناسایی می شوند

۲- زمانبندی جزئی : هر گزینه در زمانبندی ماکروسکوپی به صورت جزئی تبدیل می گردد

□ دیدگاههای زمانبندی (زمان پایان) پروژه نرم افزاری :

۱- محدودیت زمان پایان : سازمان نرم افزاری مجبور است در تاریخ خاصی نرم افزار را تحویل دهد

۲- عدم محدودیت زمان پایان ولی محدودیت‌ها در پروژه : تاریخ پایان بعد از برآورد انجام به بهترین صورت توسط تیم پروژه مشخص می شود

□ اصول موجود در زمانبندی پروژه نرم افزاری: (۷ اصل)

۱. سوا سازی (تقسیم): پروژه می بایست به تعدادی فعالیت و وظایف قابل مدیریت تقسیم گردد (تجزیه محصول و فرآیند) (فصل ۳)

۲. وابستگی متقابل (داخلی): وابستگی ها بین وظایف سوا شده بایست تعیین شوند. انواع وابستگی ها می توانند الزامات ترتیبی یا موازی باشند

۳. تخصیص زمان: زمانبندی نیاز به تخصیص واحد به هرکار دارد. هروظیفه بایست تاریخ شروع و اتمام داشته باشد و تعیین شود که اساس انجام کار پاره وقت است یا تمام وقت.

ادامه اصول زمانبندی نرم افزار از صفحه قبل

۴. **اعتبار سنجی نیروی کاری:** هر پروژه دارای تعداد معینی پرسنل است. در هنگام تخصیص زمان، مدیر پروژه بایست اطمینان حاصل کند که در هیچ زمانی تعداد بیشتری از افراد در نظر گرفته شده در برنامه کاری نباشند
مثلا اگر منابع ما ۳ نفر-روز نیرو باشد و قصد انجام ۷ کار که هر کدام ۰.۵ نفر روز فعالیت دارند پس نیرو کم می آوریم
۵. **مسئولیت های معین:** هرکاری بایست به عضوی مشخص از تیم اختصاص یابد.
۶. **نتایج معین:** هر کاری باید یک نتیجه معین داشته باشد
۷. **مراحل برجسته معین:** یک یا چندکار بایست در کنار مرحله برجسته باشد (نقطه عطف)، مرحله برجسته از لحاظ کیفی تایید شده

رابطه بین افراد و فعالیت (نیروی کار)

گاهی میتوان در پروژه های کوچک یک فرد می تواند تمام مراحل تجزیه و تحلیل، برنامه نویسی و آزمون را به تنهایی انجام دهد با افزایش پیچیدگی کار و پروژه تعداد افراد لازم افزایش می یابد. افسانه ی معروفی وجود دارد که با افزایش افراد می توان عقب افتادگی ها را جبران کرد (فصل ۱) حال آنکه افزایش تاثیر مخربی بر پروژه دارد و حتی باعث به هم خوردن برنامه زمانبندی می شود افزایش افراد باعث افزایش ارتباطات شده که ارتباطات بیشتر زمان بیشتری را در روند ساخت پروژه نیاز دارند

• مثال

چهار مهندس نرم افزار را در نظر بگیرید که هر یک قادر به تولید 5000 LOC/year می باشند، زمانی که بر روی پروژه های مستقل کار می کنند. هنگامی که این چهار مهندس در یک تیم پروژه قرار می گیرند، شش مسیر ارتباطی امکان پذیر می شود. هر مسیر ارتباطی نیازمند زمانی است که می توانست برای توسعه نرم افزار صرف شود. می توان فرض نمود که قابلیت تولید تیم (برحسب LOC) به اندازه 250 LOC/year برای هر مسیر کاهش می یابد، که سر بار ناشی از این ارتباط می باشد. بنابراین قابلیت تولید تیمی برابر است با $18500 \text{ LOC/year} = (250 \times 6) - 20000$ که 7.5 درصد کمتر از میزان مورد انتظار است پروژه ای یک ساله که این تیم بر روی آن کار می کنند از زمانبندی عقب می افتد، و با دو ماه باقیمانده، دو نفر دیگر به تیم اضافه می شوند. تعداد مسیرهای ارتباطی به 14 می رسد. قابلیت تولید افراد جدید برابر است با $1680 \text{ LOC} = 840 \times 2$ برای دو ماه باقیمانده به زمان تحویل قابلیت تولید تیمی اینک برابر است با $18180 \text{ LOC/year} = (250 \times 14) - 20000 + 1680$

❖ نکته مهم: رابطه بین افراد و کار، غیرخطی است یعنی با اضافه کردن افراد نمی توان زمان انجام کار را کاهش داد
مثلا اشتباه است اگر بگوییم کاری که با ۳ نفر، در ۲ ماه انجام می شود با ۶ نفر در ۱ ماه انجام می شود.

رابطه تجربی

اثبات رابطه غیرخطی بین زمان تکمیل پروژه و فعالیت انسانی با بکارگیری معادله نرم افزار (برای تعیین تعداد افراد و...)

معادله رابطه افراد و کارایی را بدست می آوریم

$$L = P \times E^{1/3} t^{4/3}$$

L = تعداد خطوط برنامه

E = نیروی مورد نیاز برای توسعه بر حسب نفر/ماه

P = پارامتر بهره وری است (پارامترهای برای افزایش کیفیت / مثلا ۳۰۰۰ خط در سال)

t = مدت پروژه بر حسب ماه

رابطه میزان کار و زمان (افزایش مهلت زمانی = کاهش افراد = افزایش کارایی و سود)

$$E = L^3 / (P^3 t^4)$$

E = نیروی مورد نیاز برای توسعه بر حسب نفر/سال

P = پارامتر بهره وری است

t = زمان توسعه بر حسب سال

• مهلت در مخرج قرار دارد

توزیع نیروی کار

توزیع توصیه شده برای فعالیت در طول فازهای تعریف و توسعه، قانون ۴۰-۲۰-۴۰ است، ۴۰ درصد تمام تلاشها به طرح و تجزیه و تحلیل ابتدایی و پایانی اختصاص دارد. ۴۰ درصد به آزمون نهایی مربوط است و برنامه نویسی با تاکید کمتری فقط ۲۰ درصد کل تلاش را شامل می شود.

برنامه ریزی ۲ تا ۳ درصد. تحلیل امکانات بین ۱۰ تا ۲۵ درصد و محدوده (دامنه) بین ۲۰ تا ۲۵ درصد به طرح نرم افزار مبذول می گردد
آزمایش ۳۰ تا ۴۰ درصد است ولی اگر نرم افزار حساس باشد یا انسان ارزیابی کند (ترجمه بهتر: تاثیر روی حیات انسان) زمان بیشتری به آزمایش نسبت داده می شود

تعریف یک مجموعه وظایف برای پروژه نرم افزاری

طبق مطالب بحث شده در فصل دو (فرایند) ، بر طبق فرآیند مجموعه کاری تعریف می شود
یک مجموعه واحد وظایف برای همه پروژه ها مناسب نیست. *No single set of tasks is appropriate for all projects.*
مجموعه کاری برای یک سیستم بزرگ و پیچیده برای یک سیستم ساده و کوچک افراطی است.

یک مجموعه کاری ، مجموعه ای از کارهای مهندسی نرم افزار ، هدف های اساسی (عملکردهای اصلی) و قابلیت های عرضه (تحويل) است که برای تکمیل یک پروژه خاص باید گرد هم آیند. و نباید با کار بیهوده تیم پروژه را خسته کنند

❖ طبقه بندی از انواع پروژه نرم افزاری

۱. پروژه های توسعه مفهوم : برای کشف بعضی مفاهیم تجاری یا کاربردی از یک تکنولوژی جدید (تجزیه و تحلیل) (مثال : ajax)
۲. پروژه های توسعه برنامه کاربردی جدید : با یک سفارش از طرف مشتری شروع می شود
۳. پروژه هایی افزایش برنامه کاربردی (ارتقا و اصلاح بزرگ) : اصلاحات عمده در رابطه با عملکرد که کاربرنهایی کاملاً حس می کند (مثل تغییر DB و سرعت)
۴. پروژه های نگهداری برنامه کاربردی : اصلاحات ، تصحیح تطبیق که سریعاً برای کاربر نهایی محسوس نیست (رفع یک باگ)
۵. پروژه های مهندسی مجدد : برای تجدید ساخت (ساخت مجدد) کلی یا جزئی و به علت تحلیل قبلی شناخت ساده تر است

میزان سختی و دقت

میزان دقت که با آن فرایند ساخت نرم افزار به کار گرفته می شود در مورد پروژه های مختلف متفاوت است ولی همه برای ساخت نرم افزار با کیفیت کار می کنند

❖ چهار درجه دقت متفاوت در ساخت نرم افزار قابل تعریف است :

۱. معمولی : تمام فعالیتهای چارچوب فرایند (cpf) اجرا می شوند ، فعالیتهای چتری به حداقل می رسند و مستند سازی کمتر است

❖ ادامه چهار درجه دقت متفاوت در ساخت نرم افزار قابل تعریف است:

۲. **ساخت یافته:** cpf ها (فعالیت های چارچوب فرآیند) اجرا می شوند و فعالیت های چتری متناسب با نوع پروژه انجام می شوند و تمام فعالیت های چتری مورد نیاز برای تضمین کیفیت بالا به اجرا در می آیند.

۳. **ثابت (دقیق):** درجه بالا در نظم و ترتیب اجرای cpf ها برای تضمین کیفیت بالا و تمام فعالیت های چتری نیز انجام می شوند و مستندسازی کامل صورت می پذیرد

۴. **واکنش سریع:** به خاطر سرعت فقط کارهایی به کار گرفته می شوند که به کیفیت مناسب منتهی می شوند

تعریف معیار تطابق

معیارهای تطبیق و سازگاری برای تعین میزان دقتی که با آن فرایند نرم افزاری بایست اجرا شود به کار می روند
یازده معیار سازگاری و تطابق برای پروژه های نرم افزاری تعریف شده است:

به هریک از معیارهای تطبیق شماره ای بین ۱ تا ۵ اختصاص داده می شود که ۱ نشان دهنده وجود مجموعه از کارهای فرایند و مستند سازی کم و ۵ نشان از وجود تمام کارهای فرآیند و مستند سازی زیاد است

۱. اندازه پروژه
۲. حداکثر تعداد کاربران
۳. حساسیت کار
۴. دیرپایی کاربرد (طول عمر برنامه)
۵. پایداری نیازها
۶. سهولت ارتباط مشتری/ برنامه نویس
۷. تکامل فناوری قابل کاربرد
۸. قیود و محدودیت های عملکرد
۹. خصوصیت های سیستم های جاسازی / غیر جاسازی
۱۰. اعضای تیم و کارکنان
۱۱. فاکتورهای مهندسی مجدد

محاسبه یک معیار انتخاب مجموعه وظایف

برای انتخاب مجموعه وظایف مناسب برای یک پروژه مراحل زیر را باید انجام داد:

۱. تعیین درجات ۱۱ معیار تطبیق و ثبت آنها در جدول

۲. تعیین ضریب یا میزان ارزش (عامل وزن) هر معیار بین ۰.۸ تا ۱.۲ (

۳. ضرب عامل وزن (ارزش) در ضریب نقطه ورودی (صقر یا یک) و قرار دادن در ستون حاصلضرب

شماره * عامل وزن * ضریب نقطه ورودی

۴. میانگین داده های ستون حاصلضرب را حساب کرده و نتیجه را به عنوان انتخابگر مجموعه وظایف (TSS) می نویسیم

ضریب امتیاز ورودی

پروژه های توسعه کاربرد پیشرفت نگهداری مهندسی پروژه های

مهندسی		تکنولوژی		پشتیبانی		کاربرد		توسعه			
محصول		نقطه ورودی				ارزش		درجه		معیار انطباق	
		Reeng.	NDev.	Enhanc.	Maint.	Conc.					
_____		1	1	1	1	0	1.20	_____	اندازه پروژه		
_____		1	1	1	1	0	1.10	_____	تعداد کاربران		
_____		1	1	1	1	0	1.10	_____	بحرانی بودن تجارت		
_____		0	1	1	0	0	0.90	_____	طولانی مدت		
_____		1	1	1	1	0	1.20	_____	پایداری نیازها		
_____		1	1	1	1	1	0.90	_____	سهولت ارتباط		
_____		1	1	0	0	1	0.90	_____	کامل بودن تکنولوژی		
_____		1	1	1	0	0	0.80	_____	محدودیت های اجرا		
_____		1	1	1	0	1	1.20	_____	جاسازی شده / جاسازی نشده		
_____		1	1	1	1	1	1.00	_____	مدیریت پروژه		
_____		1	1	1	1	0	1.10	_____	عملکرد داخلی		
_____		1	0	0	0	0	1.20	_____	فاکتورهای مهندسی مجدد		

task set selector

تفسیر مقدار انتخاب مجموعه وظایف (TSS)

- اتفاقی (معمولی) $TSS < 1.2$
- ساختاری (ساخت یافته) $1.0 < TSS < 3.0$
- ثابت $TSS > 2.4$

انتخاب وظایف مهندسی نرم افزار (بعد از تعیین دقت)

برای داشتن یک برنامه زمانبندی پروژه، مجموعه وظایف باید در مدت زمانی پروژه توزیع گردد. مجموعه وظایف نسبت به نوع پروژه و میزان دقت تغییر خواهد کرد. هر یک از پروژه های توصیف شده در بخش قبل، ممکن است با به کاربردن یک **مدل فرایندی** که خطی، متوالی و تکراری (مثلا مدل های افزایشی یا نمونه سازی) و یا تکاملی است (مانند مدل حلزونی) قابل دسترسی باشند، در بعضی موارد یک پروژه به آرامی به پروژه بعدی ملحق میشود. مثلاً ممکن است پروژه توسعه مفهوم به پروژه توسعه کاربرد جدید تکامل یابد. یا پس از ایجاد پروژه توسعه کاربرد ممکن است پروژه افزایش کاربرد شروع شود. این پیشرفت طبیعی و قابل پیشگویی است و بدون توجه به مدل فرایندی که توسط یک سازمان در پیش گرفته شده است اتفاق می افتد. بنابراین وظایف مهندسی نرم افزار گفته شده در بخش هایی بعدی، در مورد تمام مدل های فرایندی قابل اجرا هستند

❖ (مثال) وظایف مهندسی نرم افزار برای یک پروژه توسعه مفهوم: (اطمینان از قابل دستیابی بودن)

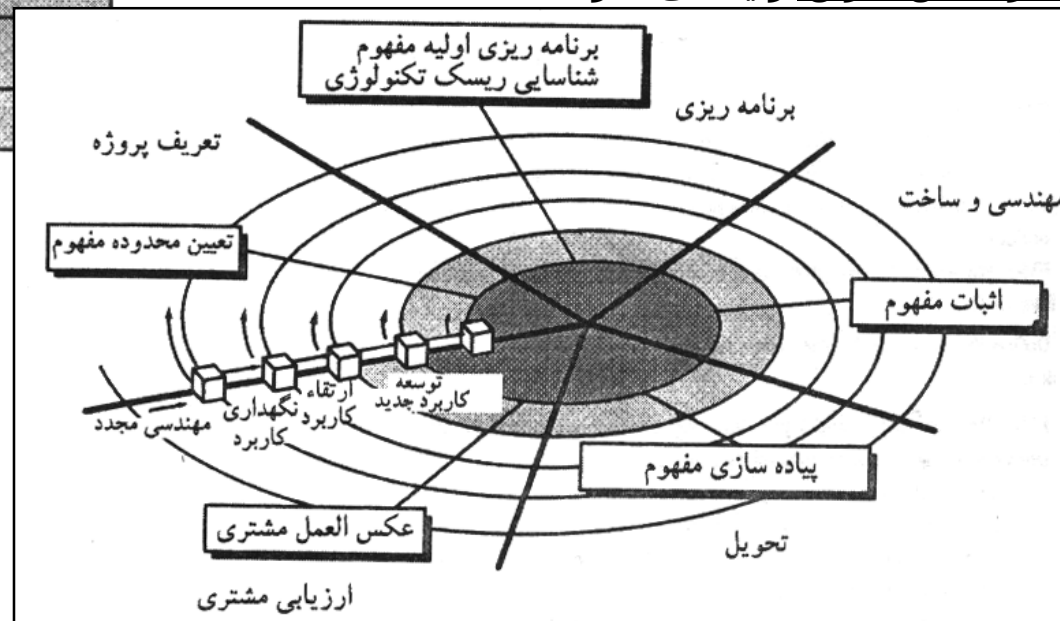
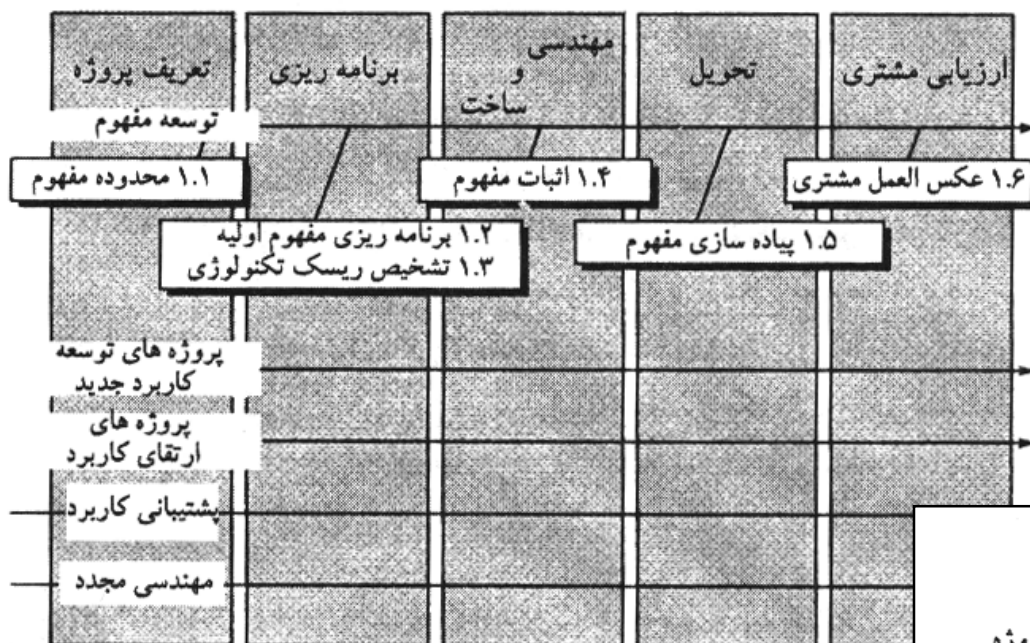
- **تعیین دامنه مفهوم** - توانایی سازمان برای انجام کاری که در دامنه پروژه نهفته است را به وجود می آورد
 - **طرح ریزی اولیه مفهوم** - برقرار ساختن توانایی سازمان در به عهده گرفتن کارهای مربوط به حوزه پروژه
 - **ارزیابی ریسک فنی** - خطری را که همراه فناوری مورد کاربرد حوزه پروژه است را ارزیابی می کند
 - **آزمون (و اثبات) مفهوم** - کارایی و عملی بودن یک فناوری نو را حوزه نرم افزاری نشان می دهد
 - **پیاده سازی مفهوم** - نمود مفهوم را به گونه ای که توسط یک مشتری قابل بررسی باشد پیاده می کند و برای اهداف بازیابی به کار می رود ، زمانی که در نظر باشد مفهومی به مشتریان با مدیریت دیگری فروخته شود
 - **واکنش مشتری به مفهوم** - واکنش نسبت به مفهوم یک فناوری نو را جستجو می کند و کاربردهای مشتری خاصی را مورد هدف قرار می دهد.
- حرکت مهندسی نرم افزار به سوی پروژه های توسعه مفهوم (و نیز تمام انواع دیگر پروژه ها) تا اندازه کمی از عقل سلیم بیشتر است. تیم نرم افزار باید بداند که چه کارهایی باید انجام گیرد (تعیین حوزه) تیم (یا مدیر) باید تعیین کند که آیا کسی برای انجام آن وجود دارد (برنامه ریزی) خطرات ممکن را مورد ملاحظه قرار دهد (تخمین خطر) فناوری را تا اندازه ای مورد تامین قرار دهد (آزمون مفهوم) و آن را به گونه ای شاخص پیاده نماید به طوری که مشتری بتواند آن را ارزیابی کند (پیاده سازی مفهوم و ارزیابی مشتری) و در پایان ، اگر مفهوم کارآمد است، یک گونه تولیدی (تبدیلی) باید تولید گردد.

• به کارگیری مجموعه وظایف مهندسی در دو مدل فرایند متفاوت

فعالیت های شبکه وظائف توسعه مفهوم مایهتا تکراری است. یعنی : یک پروژه توسعه مفهوم واقعی از راه تعدادی افزایش برنامه ریزی شده به این فعالیت ها نزدیک می شود که هریک مختص تولید یک محصول قابل تحویل، که توسط مشتری قابل ارزشیابی است، هستند.

اگر یک مدل فرایند خطی انتخاب شده باشد، هر یک از این افزایشها در یک توالی تکراری تعریف شده اند.

در خلال هر مرحله، فعالیتهای پوششی (چتری) اجرا می شوند. کیفیت کنترل می شود و در پایان هر مرحله یک محصول قابل تحویل تولید می شود.



پالایش وظایف اصلی

• کارهای عمده که در بخش قبل توصیف گردیدند ممکن است برای تعریف یک برنامه زمان بندی **ماکروسکوپی** برای یک پروژه به کار روند. با این وجود، برنامه زمان بندی ماکروسکوپی برای ایجاد یک **برنامه جزئی** برای پروژه باید **پالایش** گردد. پالایش با در نظر گرفتن کارهای بزرگ و تجزیه آنها به کارهای فرعی (با محصولات کاری و هدفهای اصلی مربوط) شروع می شود.

❖ **مثال:** به عنوان نمونه ای از تجزیه کردن وظیفه، تعیین حوزه برای یک پروژه توسعه مفهوم [*] را، که در بخش قبل بحث شد، مورد ملاحظه قرار دهید. پالایش وظیفه با بکار بردن یک طرح خلاصه شده قابل انجام است، اما در این کتاب، یک رهیافت زبان طراحی فرایندی برای نشان دادن جریان فعالیت تعین حوزه مفهوم به کار برده می شود:

۱-۱-۱ تعیین حوزه مفهوم

۱-۱-۱-۱ نیازها، منافع و مشتریان بالقوه را شناسایی کنید.

۱-۱-۱-۲ رویدادهای برونی کنترل و درونی را که کاربردها را هدایت می کنند، تعریف کنید.

وظیفه ۱-۱-۲ را شروع کنید.

۱-۲-۱-۱ FTR: (بازبینی رسمی فنی) توصیف نوشتاری نیازها را بررسی کنید.

۱-۲-۱-۲ لیستی از خروجی ها / ورودیهای قابل مشاهده مشتری بدست دهد.

بر حسب مورد : دستگاهها

اگر دستگاه ها= کیفیت و آمادگی کارکردی

با مشتری ملاقات نمایید تا نیازهای مفهومی اصلی مشخص گردند. با مصرف کنندگان نهایی مصاحبه کنید

رهیافت کنونی به مسئله . روند جاری را مورد مشاهده قرار دهید تقاضاها و شکایات گذشته را بررسی کنید

اگر دستگاه ها= تجزیه و تحلیل ساخت یافته.

لیستی از اشیا داده های اصلی تهیه کنید رابطه بین اشیا را تعریف کنید ویژگی اشیا را تعریف کنید

اگر دستگاه ها : دید عینی

لیستی از کلاسهای مسائل تهیه کنید سلسله مراتب کلاس و ارتباط کلاس را توسعه دهید ویژگی های کلاسها را توصیف کنید

پایان مورد

۱-۲-۱-۲ ورودی/ خروجی ها را با مشتری مرور کنید و در موارد لازم آنها را اصلاح کنید

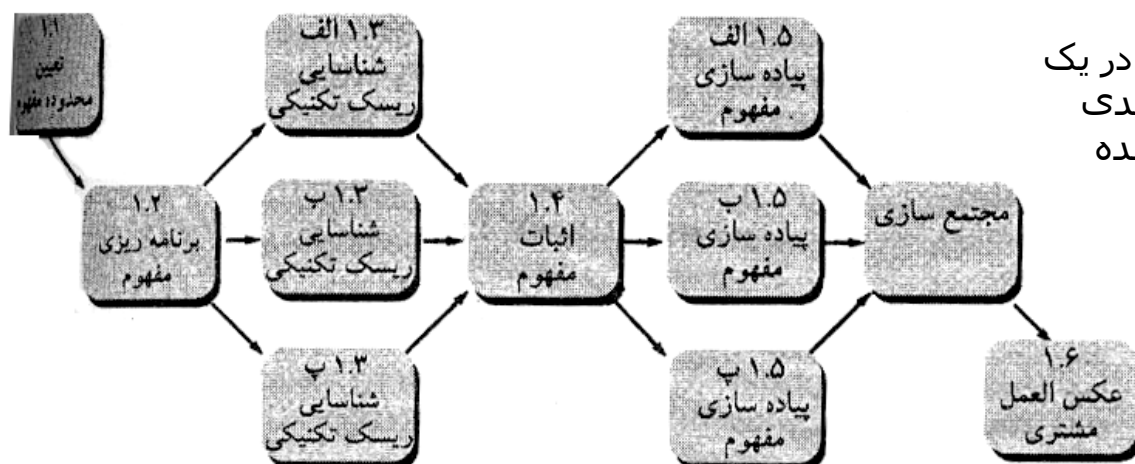
تعریف یک شبکه وظایف

وظائف و وظائف فرعی بر پایه **ترتیبشان**، به هم وابستگی های متقابل دارند. اگر بیش از یک نفر در یک پروژه نرم افزاری باشند، احتمالاً فعالیتها و وظائف توسعه ای به صورت **موازی** اجرا می شوند پس کارهای **همروند** باید با هم **هماهنگ** شوند، طوری که وقتی کارهای بعدی تولید کاری آنها را نیاز دارد آنها کامل شده باشند.

یک شبکه وظائف (= شبکه فعالیت)، **نمایشی نموداری و گرافیکی از جریان وظیفه** برای پروژه می باشد. گاهی از آن به عنوان **مکانیسمی برای انتقال توالی وظیفه و وابستگی های** آن به یک ابزار برنامه ریزی خودکار پروژه، استفاده می شود.

• در ساده ترین شکل آن برای ایجاد یک برنامه ماکروسکوپی به کار می رود، شبکه وظیفه در توصیف کارهای اصلی مهندسی نرم افزار به کار می رود.

ماهیت همزمانی فعالیتهای مهندسی نرم افزاری منجر به **نیازمندیهای برنامه ریزی مهمی** می شود. از آنجا که کارهای موازی همزمان اتفاق می افتد، برنامه ریز باید **وابستگی های بین کارها** را مشخص نماید تا پیشرفت مداوم در جهت تکمیل کار تضمین گردد به علاوه، **مدیر پروژه** باید **آگاهی** از وظائفی که در **موقعیت بحرانی** قرار می گیرند داشته باشد. یعنی، آن کارهایی که در برنامه زمانبندی باید کامل گردند



• **مثال:** شبکه وظایف در تصویر مقابل ماکروسکوپی است. در یک شبکه وظایف جزئی (که مقدمه ای برای یک برنامه زمان بندی جزئی است) هر فعالیتی که در تصویر مقابل نشان داده شده است باید بسط داده شود



زمانبندی

زمانبندی پروژه توزیع زمانی مجموعه فعالیتها در طول مدت پروژه که بسته به پروژه نرم افزاری تغییر می کنند. زمانبندی پروژه نرم افزاری تفاوت چندانی با زمانبندی فعالیتهای کاری دیگر مهندسی نرم افزار ندارد. پس با کمی اصلاح ابزارهای زمانبندی عمومی پروژه ها و نیز فنون آن در مورد پروژه های نرم افزاری نیز قابل اجرا هستند.

۱- ارزشیابی برنامه و فنون بررسی (PERT) و ۲- روش مسیر بحرانی (CPM) دو روشی هستند که برای زمانبندی پروژه به کار می روند که برای توسعه نرم افزاری نیز قابل کاربردند.

❖ این دو تکنیک به وسیله اطلاعات حاصل از فعالیتهای برنامه ریزی پروژه قبلی بدست آمدند، یعنی قبل از زمانبندی بایست:

- برآوردهای نیروی انسانی
- تجزیه کارکرد محصول
- انتخاب مدل فرآیند مناسب و مجموعه وظائف
- تجزیه وظائف

با به کار بردن یک شبکه وظائف امکان تعریف وابستگی های موجود بین وظائف وجود دارد. وظائف که بعضی وقتها به آنها ساختار تجزیه کار پروژه (WBS) گفته می شود، برای کل پروژه یا کارکردهای فردی تعریف می شوند

❖ PERT و CPM به برنامه ریز نرم افزاری امکان می دهد:

۱. مسیر بحرانی را مشخص نماید - زنجیره وظائفی که مدت زمانی پروژه را تعیین می کند.
۲. "محتمل ترین" برآورد زمانی برای کارهای فردی را با به کار بردن مدلهای آماری انجام دهد.
۳. زمانهای مرزی را محاسبه کند که به تعریف یک "پنجره" زمانی برای یک وظیفه خاص می انجامد. (۵ نوع زمان مرزی)

محاسبه زمانهای مرزی پروژه نرم افزاری می تواند بسیار مفید باشد، مثلاً: سهل انگاری در طرح یک تابع می تواند توسعه بیشتر توابع دیگر را به تاخیر اندازد.

Critical path method (CPM)

Program evaluation and review technique (PERT)

Work Breakdown Structure (WBS)

□ زمانهای مرزی مهمی که ممکن است از CPM یا PERT بدست آیند:

- (۱) زودترین زمانی که يك وظیفه می تواند شروع شود (در صورت تکمیل کارهای قبلی)
- (۲) دیرترین زمان برای شروع وظیفه قبل را به تاخیر افتادن زمان حداقل تکمیل پروژه.
- (۳) زودترین زمان خاتمه وظیفه = زودترین زمان شروع + مدت زمانی وظیفه
- (۴) دیرترین زمان خاتمه وظیفه = دیرترین زمان شروع + مدت زمانی وظیفه
- (۵) تعلیق کلی (زمان شناوری): مقدار زمان اضافی یا میزان مجاز تغییرات زمانی، طوریکه مسیر بحرانی پروژه بر طبق برنامه باقی بماند.

محاسبه مرزی منجر به تعیین مسیر بحرانی می شود و روشی کمی در اختیار مدیر می گذارد که با آن می تواند به هنگام تکمیل شدن کارها پیشرفت آنها را ارزشیابی کند دو روش CPM و PERT را با نرم افزارهای زیادی می توان پیاده سازی کرد

نمودارهای خطی زمانی

هنگام ایجاد یک برنامه زمان بندی برای پروژه نرم افزاری برنامه ریز آن، کارش را با مجموعه ای از وظائف (ساختار تقسیم کار) شروع می کند. اگر ابزارهای خودکار به کار روند، تقسیم کار به عنوان شبکه وظیفه یا خلاصه وظیفه در نظر گرفته می شود. **نیروی کاری**، **مدت زمانی**، و **زمان شروع وظیفه و ورودی هر وظیفه** در نظر گرفته می شود. همچنین ممکن است وظائف با افراد خاصی اختصاص داده شوند.

در نتیجه این کار، یک نمودار زمانی (نمودار گانت) به وجود می آید. یک نمودار زمانی را می توان به کل پروژه توسعه داد. از طرف دیگر، می توان نمودارهای جداگانه نیز برای هر یک از توابع پروژه یا برای کارهای فردی که بر روی پروژه انجام می شوند به وجود آورد.



نام درس: مهندسی نرم افزار ۱

تهیه کننده: ایمان مختاری

مدرس دانشگاه‌های پیام نور شهر کرد-فارسان-فرخ شهر

[Http://www.Snu.ir](http://www.Snu.ir)

شماره صفحه : ۱۰۶



- تمام کارهای پروژه (برای تعیین حوزه مفهومی) در ستون سمت چپ لیست گردیده اند.
- باریکه های افقی مدت زمانی هر وظیفه را نشان می دهند. وقتی چندین باریکه در یک زمان اتفاق می افتند، نشان دهنده همزمانی کارهاست.
- لوزی ها نشان دهنده نقاط عطف اصلی اند.

یک بار اطلاعات لازم برای به وجود آوردن یک نمودار زمانی وارد گردیده است و اکثر ابزارهای زمانبندی پروژه های نرم افزاری **جدولهای پروژه** ای تولید کرده اند - که لیستی جدول وار از تمام وظائف پروژه، شروع برنامه ریزی شده و واقعی و زمانهای پایان کار، و انواعی از اطلاعات مربوطه میباشد.

جدولهای پروژه که به همراه جدولهای زمانی به کار می روند به **مدیر پروژه امکان پیگیری پیشرفت کار** را می دهد

نیازها	فعالیت اختصاص یافته	افراد اختصاص یافته	تکمیل واقعی	تکمیل طبق برنامه	شروع واقعی	شروع برنامه ریزی شده	کارها
Scoping will require more effort/time	2 p-d	BLS	wk1, d2	wk1, d2	wk1, d1	wk1, d1	۱.۱.۱ مشخص نمودن نیازها و فواید
	1 p-d	JPP	wk1, d2	wk1, d2	wk1, d2	wk1, d2	ملاقات با مشتری
	1 p-d	BLS/JPP	wk1, d3	wk1, d3	wk1, d3	wk1, d3	مشخص نمودن نیازها و محدودیت های پروژه
			wk1, d3	wk1, d3	wk1, d3	wk1, d3	نشانگر: تعریف شدن حکم محصول
							۱.۱.۲ تعریف خروجی - کنترل - ورودی مطلوب (OCI)
	1.5 p-d	BLS		wk2, d2	wk1, d4	wk1, d4	محدوده توابع صفحه کلید
	2 p-d	JPP		wk2, d2	wk1, d3	wk1, d3	محدوده توابع ورود صوت
	1 p-d	MLL		wk2, d3		wk2, d1	محدوده مود ارتباط
	1.5 p-d	BLS		wk2, d2		wk2, d1	محدوده شناسایی سند
	2 p-d	JPP		wk2, d3	wk1, d4	wk1, d4	محدوده توابع دیگر WP
	3 p-d	MLL		wk2, d3		wk2, d1	مستند سازی
	3 p-d	all		wk2, d3		wk2, d3	FTR: مرور OCI با مشتری
	3 p-d	all		wk2, d4		wk2, d4	معیار: تعریف OCI
				wk2, d5		wk2, d5	۱.۱.۳ تعریف عملکرد - رفتار



ردگیری برنامه زمانبندی

برنامه ی زمانبندی پروژه : وظایف و نقاط عطف اصلی ای را که باید در جریان پیشرفت پروژه پیگیری و کنترل شوند، تعریف می کند.

• پیگیری به طور خلاصه به روشهای زیر صورت می پذیرد:

- ۱- برگزاری گردهمایی
 - ۲- ارزشیابی نتایج تمام بررسی ها
 - ۳- مقایسه زمان تحقق یافتن اهداف رسمی (نقاط عطف) و زمان برنامه ریزی شده
 - ۴- مقایسه ی شروع واقعی برای هر وظیفه و شروع برنامه ریزی شده (در گانت)
 - ۵- ملاقات غیر رسمی با کارورزان به منظور دریافت میزان پیشرفت یا خطرات
 - ۶- محاسبه ی ارزیابی کمی پیشرفتها با بکار گیری ارزش سنجی اکتسابی
- ### • وظایف مدیر پروژه هنگام بروز مشکل:

به اعمال کنترل می پردازد. در این هنگام منابع بیشتری به آنها اختصاص داده می شود و ترتیب و استخدام کارکنان تغییر می کند و یا برنامه ی پروژه ی زمانبندی اصلاح می شود.

• [Time-boxing]: جعبه ای کردن زمان (تلاش جهت رسیدن به هر وظیفه فقط در محدوده خودش)

در مواردی که فشار زمانی حاد به وجود می آید، مدیران با تجربه اقدام به بکارگیری برنامه ی زمانبندی و تکنیک کنترل خاص در برنامه می کنند که جعبه ای کردن زمان نام دارد

راهبرد جعبه ای کردن زمان تشخیص می دهد که کل محصول کامل در موعد مقرر قابل عرضه نخواهد بود لذا یا مدل فرایند افزایشی انتخاب می شود و برای هر مرحله افزایش برنامه ای تدوین می شود.

وظیفه ای که به هر یک از مراحل افزایشی اختصاص داده شده است از نظر زمانی به اصطلاح جعبه ای شده است.

برنامه ی زمانی، برای هر وظیفه تنظیم شده است و این کار با برگشت از زمان تحویل محصول به مراحل افزایشی انجام گرفته است. یک جعبه در اطراف هر وظیفه گذاشته شده است. هرگاه وظیفه ای به مرز زمان جعبه اش می رسد، آن وظیفه متوقف می شود و وظیفه ی بعدی آغاز می شود. (با تفرانس ۱۰ درصد انجام شده پس مابقی برای بعد)



تحلیل مقدار بدست آمده (ارزش سنجی اکتسابی):

❖ ارزش سنجی اکتسابی

تکنیکی که برای تحلیل کمی پیشرفت وجود دارد، ارزش سنجی اکتسابی (EVA) نامیده می شود
❖ نظر فلیمنگ و کاپلمن در این مورد (مقادیر بدست آمده و اکتسابی):

این ارزشیابی برداشت دقیق و قابل اطمینانی از اجرای پروژه با گذشت ۱۵٪ اولیه ی آن به بعد فراهم می کند.
❖ تعیین ارزش اکتسابی طبق مراحل زیر است:

۱- ارزش بودجه ای کار زمانبندی شده (BCWS) برای هر وظیفه ای که در برنامه است، تعیین می شود.

۲- مقدارهای ارزش بودجه ای کار زمانبندی شده (یعنی: مقادیر BCWS) برای تمام کارها جمع شده اند، تا بودجه را در زمان تکمیل پروژه (BAC) نشان دهند.
$$BAC = \sum (BCWS_k) \text{ for all tasks } k$$

۳- مرحله ی بعد، ارزش بودجه ای کار اجرا شده (BCWP) محاسبه می شود. این مقدار برابر است با حاصل جمع مقدارهای ارزش بودجه ای کار اجرا شده برای همه ی کارهایی که واقعا در یک نقطه ی زمانی در زمانبندی پروژه ی تکمیل شده است.

• تفاوت بین ارزش بودجه ای کار زمان بندی شده و ارزش بودجه ای کار اجرا شده در این است که اولی نماینده ی بودجه ی فعالیتهایی است که برای تکمیل آنها برنامه ریزی شده است، در حالی که دومی نماینده ی فعالیت هایی است که واقعا کامل شده اند

Earned value analysis (EVA)

budgeted cost of work scheduled (BCWS)

budget at completion (BAC)

budgeted cost of work performed (BCWP)



با داشتن مقدار های ارزش بودجه ای کار زمان بندی شده، بودجه ی "تکمیل کار و ارزش بودجه ای کار اجرا شده، نشانگر های عمده ی پیشرفت را می توان محاسبه کرد.

Schedule Performance Index , $SPI = BCWP/BCWS$

ارزش بودجه ای کار زمان بندی شده/ارزش بودجه ای کار اجرا شده= شاخص عملکرد زمانبندی

Schedule variance, $SV = BCWP - BCWS$

ارزش بودجه ای کار زمان بندی شده-ارزش بودجه ای کار اجرا شده= واریانس زمانبندی

Percent Schedule for completion = $BCWS/BAC$

بودجه به هنگام تکمیل کار/ارزش بودجه ای کار اجرا شده = درصد برنامه ریزی شده برای تکمیل کار که نشانگر کمی از کامل شدن پروژه است

Cost Performance Index , $CPI = BCWP/ACWP$

ارزش واقعی کار اجرا شده/ارزش بودجه ای کار اجرا شده= شاخص اجرایی هزینه

Cost Variance, $Cv = BCWP - ACWP$

ارزش واقعی کار اجرا شده-ارزش بودجه ای کار اجرا شده= واریانس هزینه

• $ACWP$ = هزینه واقعی کار انجام شده

رد گیری خطا

جستجوی خطاها به شما اجازه می دهد، فعالیت کنونی خود را با نیروی پیشین مقایسه نموده و یک معیار کمی برای کیفیت کاری که اداره می شود، تهیه کنید.

• تیم نرم افزاری به اجرای بررسی های فنی رسمی (و سپس آزمودن) برای یافتن و اصلاح خطاهای E، که در محصولات کاری در خلال وظایفشان تولید شده اند می پردازند.
هر خطایی که کشف نشود به عنوان نقص در نظر گرفته می شود.
D کارایی و بازده رفع نقص به صورت زیر تعریف می شود.

$$D=E/(E+D)$$

• Dre يك متریک فرآیند می باشد که نمایش قوی از موثر بودن فعالیتهای **تضمین کیفیت** را فراهم میکند و همچنین می تواند توسط مدیر پروژه استفاده شود تا او را در تعیین پیشرفت پروژه در کارهای زمانبندی حمایت کند. مدیر پروژه بر حسب DRE می تواند تصمیم بگیرد که چه قسمتهایی نیاز به مرور دارند که اینکار باعث صرفه جویی در زمان می شود

طرح (برنامه) پروژه

هر مرحله در مهندسی نرم افزار بایست عنصر قابل تحویلی را تولید کند، با اعمال برنامه ریزی طرح پروژه نرم افزاری به وجود آمده است. که شامل اطلاعات مبنایی برای هزینه و زمانبندی است.

خصوصیات برنامه پروژه نرم افزاری:

۱- بیان حوزه و منابع برای مدبر، سرپرست و شمتري

۲- تعریف ريسك و راههای کاهش

۳- تعریف هزینه و زمانبندی

۴- تعیین روش کلی ساخت نرم افزار

۵- تعیین نحوه تضمین کیفیت و مدیریت تغییرات

طرح پروژه ایستا نیست و در مراحل مختلف توسط تیم پروژه به روز می شود

فصل ۸: تضمین کیفیت نرم افزار (SQA)

Software quality assurance (SQA)

فن مدبران

آز مساس

تضمین کیفیت نرم افزار چیست؟

فعالیتی چتری پوششی است که در سراسر فرآیند به کار گرفته می شود و شامل (۱) روش مدیریت کیفیت (۲) فناوری مهندسی نرم افزار (روشها/ابزارها) (۳) بررسیهای فنی رسمی در طول اجرای فرایند (۴) روش آزمایش چند لایه ای (۵) کنترل مستندات و تغییرات انجام شده روی آنها (۶) روالی برای تایید انطباق با استانداردهای توسعه نرم افزار (۷) روشهای اندازه گیری و گزارش

تضمین کیفیت نرم افزار در مورد چه چیزی بحث می کند؟

تنها گفتن اینکه کیفیت نرم افزاری مهم است کافی نمی باشد--- (۱) باید به روشنی بیان کنید که وقتی میگویید "کیفیت نرم افزار" منظورتان چیست--- (۲) ایجاد فعالیت هایی را که به تضمین کیفیت بالای هر محصول مهندسی نرم افزاری کمک می کنند--- (۳) اجرای فعالیت های تضمین کیفیت را بر روی هر پروژه نرم افزاری -- (۴) استفاده از متریک هایی که استراتژی هایی را برای ارتقا فرآیند و در نتیجه ارتقا کیفیت محصول نهایی توسعه دهند

چه کسی آنرا انجام می دهد؟

همه افراد در فرآیند مهندسی نرم افزار است ، مسئول کیفیت اند.

دلیل اهمیت تضمین کیفیت؟

اگر تیم نرم افزار در تمام فعالیتهای مهندسی نرم افزار بر کیفیت تاکید داشته باشد، میزان کار محدودی که باید انجام شود کاهش می یابد. یعنی نیاز به دوباره کاری نیست و کاهش هزینه ها و صرفه جویی در زمان را به دنبال دارد

مراحل آن چیست؟

۱ تعریف کیفیت نرم افزار در سطح انتزاعی ، پس از درک مفهوم کیفیت در يك پروژه ۲ تعیین فعالیتهای SQA برای حذف به موقع خطاها

حاصل کار چیست؟

يك طرح تضمین کیفیت برای تعریف راهبرد SQA به وجود می آید.

در ضمن تجزیه تحلیل و کدنویسی ؛ محصول کار اولیه SQA عبارت است از : گزارش خلاصه مرور تکنیکی

در خلال آزمون ، طرحها و رويه های آزمون تولید می شوند

چگونه از صحت انجام آن می توان مطمئن شد؟

۱ یافتن خطاها ، قبل از آنکه به اشکالات عمده تبدیل شوند و ۲ کاهش میزان دوباره کاری

مفاهیم کیفیت

پدیده گوناگونی بین نمونه ها ، همانند تفاوت دو دانه ی برف ، برای تمام محصولات ساخت انسان نیز صادق است حتی دو برد الکترونیکی اگر با دقت (ذره بین) بررسی شوند متوجه تفاوت در اندازه سیمهای مسی و ... می شویم

کنترل تنوع و گوناگونی (تفاوت) قسمت اصلی کنترل کیفیت است. یک تولیدکننده حتی وقتی به کاری ساده چون کپی کردن دیسک میپردازد تلاش دارد گوناگونی بین تولیدات به حداقل برساند اما ممکن است به دلیل تنوع بین نمونه ها با مشکل مواجه شود.

در هر پروژه ای نسبت به پروژه دیگر، ما تمایل داریم **تفاوت بین منابع پیش بینی شده برای کامل کردن یک پروژه و منابع واقعی** ای که به کار برده شده اند شامل (افراد تیم ، تجهیزات و تقویم زمانی) را به **حداقل برسانیم**

ما نه تنها می خواهیم تعداد نقایصی را که وارد محیط ما شده اند به حداقل برسانیم بلکه تمایل داریم مطمئن شویم که **واریانس تعداد اشکالات نیز در هر نشر نسبت به دیگری به حداقل رسیده است.** هدف به حداقل رساندن تفاوت ها در سرعت ، دقت در پاسخگویی به مشکلات مشتری است.

کیفیت

کیفیت به عنوان خصیصه چیزی به ویژگیهای قابل اندازه گیری چیزی اشاره دارد .

ویژگیهایی که ما می توانیم آن ها را با استاندارد های شناخته شده ای چون طول رنگ و غیره **مقایسه** کنیم. علی رغم فکری بودن نرم افزار مقیاس هایی برای ویژگیهای یک برنامه هم وجود دارد. این ویژگیها عبارتند از: پیچیدگی ذاتی ، یکپارچگی تعداد امتیازات کارکردی خطوط برنامه نویسی و...

وقتی ما چیزی را بر مبنای ویژگیهای قابل سنجش (اندازه گیری) آن می آزمایشیم ممکن است با دو نوع کیفیت رو به رو شویم:

(۱) کیفیت طراحی و (۲) کیفیت تطابق.

کیفیت طراحی مربوط به آن ویژگیهای است که طراح به چیزی اختصاص داده است. (درجه مواد، تلرانس ها ، مشخصات کارایی و...) **کیفیت تطابق** میزان رعایت مشخصات طراحی به هنگام ساخت محصول است، بالاتر بودن درجه تطابق سبب کیفیت بالاتر می شود

در توسعه نرم افزار ،

کیفیت طراحی : نیازها، مشخصه ها و طراحی سیستم* **کیفیت تطابق** ابتدا بر پیاده سازی تمرکز دارد، اگر در پیاده سازی کیفیت طراحی رعایت شود و بر اساس نیازها منطبق شود کیفیت تطابق بالا می رود. عنصر مهم دیگری که برای مهندس نرم افزار بایست در کنار کیفیت طراحی و کیفیت تطابق مهم باشد، **رضایت مشتری** است یعنی:

رضایت مصرف کننده = محصول مورد قبول + کیفیت خوب + تحویل طبق بودجه و زمانبندی

یعنی " کیفیت يك محصول تابعي است از میزان تغییری که آن محصول در بهتر شدن جهان انجام می دهد."

کنترل کیفیت

کنترل تغییر ممکن است با **کنترل کیفیت** برابر گردد. کنترل کیفیت عبارت است از **مجموعه بازرسیها بررسی و آزمون هایی** که در روند فرایند نرم افزاری انجام میگردد و هدف آن تضمین تطبیق تولید کالای با نیازهایی است که به خاطر آن تولید می شود. کنترل کیفیت شامل يك **حلقه بازگشتی** است که محصول نهایی را تولید می کند.

ترکیب سنجش (اندازه گیری) + فیدبک باعث می شود که اگر خصوصیتهاي محصول برآورد نشد، به صورت بازگشتی فرایند ایجاد آنها تنظیم شوند در اینصورت **کنترل کیفیت بخشی از فرایند تولید** است این فعالیتها را می توان دستی، خودکار یا ترکیبی از هر دو انجام داد

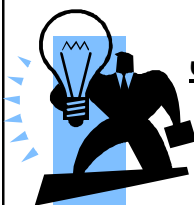
تضمین کیفیت

تضمین کیفیت شامل حسابرسی و گزارش کارکردهای مدیریت است. هدف تضمین کیفیت **فراهم کردن اطلاعات لازم برای مدیریت به منظور اطلاع از کیفیت محصول و در نتیجه اطمینان کامل که کیفیت تولید در جهت برآوردن اهداف است.**

اگر داده های حاصل از تضمین کیفیت مشکلات را آشکار کنند ،وظیفه مدیر به **کارگیری منابع برای حل مسائل** است

هزینه کیفیت

هزینه کیفیت شامل تمام هزینه هایی است که در جهت نیل به کیفیت با انجام فعالیت های مربوط به کیفیت است. مطالعات هزینه کیفیت جهت فراهم کردن بستری برای هزینه کیفیت فعلی شناسایی کنند و راههایی برای کاهش هزینه کیفیت و فراهم کردن یک اساس طبیعی برای مقایسه می باشد (اکثرا دلان).



هزینه های کیفیت ممکن است به هزینه هایی که مربوط به ^۱پیشگیری ، ^۲ارزیابی و ^۳شکست میباشد تقسیم کرد.
هزینه های **پیشگیری** عبارتند از:

- برنامه ریزی کیفیت
- تجهیزات آزمون
- بررسیهای منظم فنی
- آموزش

هزینه های **ارزیابی** شامل فعالیت هایی برای آگاهی از وضعیت تولید در آغاز هر فرایند میباشد. مثالهایی از هزینه های ارزیابی:

- بازرسی درون-مرحله ای و بین-مرحله ای
- تعمیر و نگهداری تجهیزات
- انجام آزمون

• هزینه های **شکست** در صورت معلوم نشدن نقصی قبل از تحویل محصول به مشتریان از بین می روند. هزینه های شکست ممکن است به هزینه های **شکست درونی** و هزینه های **شکست بیرونی** تقسیم گردند .

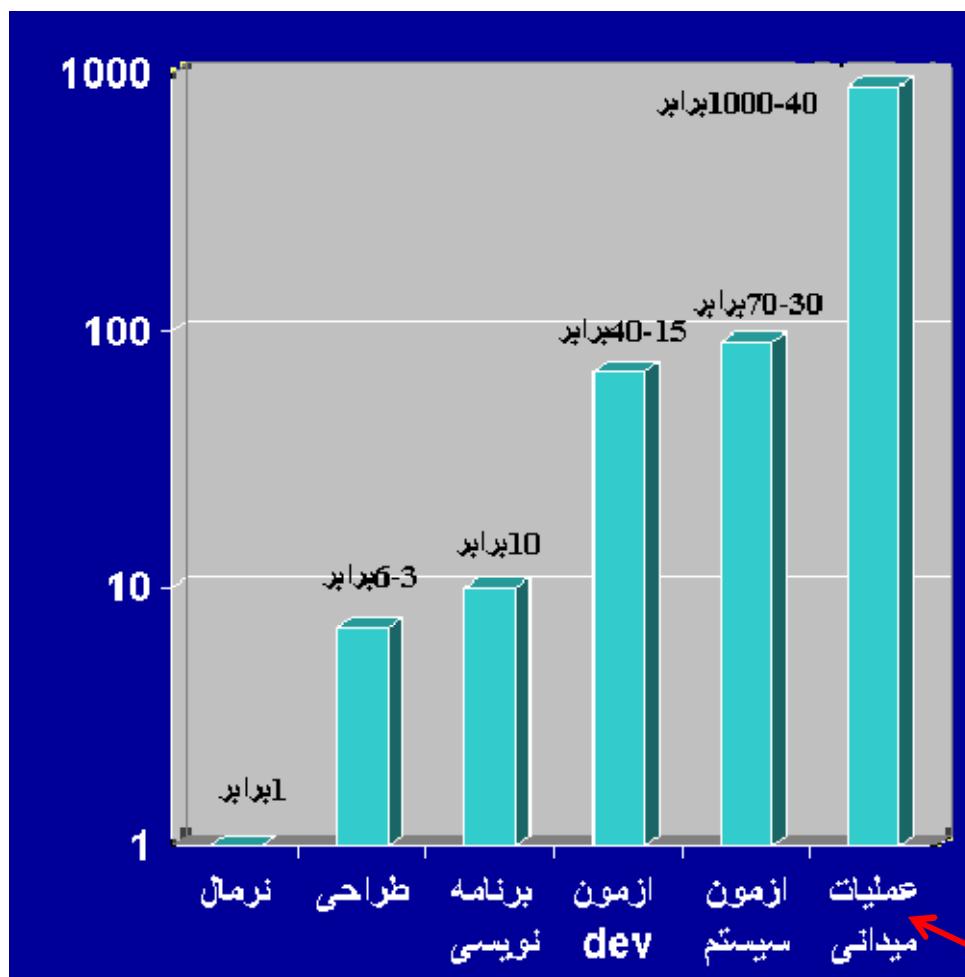
• هزینه های شکست داخلی عبارتند از:

(۱) دوباره کاری (۲) تعمیر (۳) تجزیه و تحلیل وضعیت شکست

• هزینه های شکست بیرونی انهایی هست که مربوط به نقایصی هستند که بعد از ارسال محصول به مشتری پیدا شده اند. نمونه هایی از هزینه های شکست بیرونی عبارتند از:

- هزینه رفع و رجوع شکایات و دعاوی حقوقی
- بازگشت و تعویض
- پشتیبانی و کمک های حمایتی
- انجام تعهدات تضمین داده شده

• همان طور که انتظار می رفت هزینه نسبی پیدا کردن و رفع یک نقص با حرکت ماز پیشگیری به تشخیص و سپس به هزینه های شکست داخلی و شکست خارجی به میزان قابل ملاحظه ای افزایش می یابد.



نمودار هزینه نسبی اصلاح خطا

• مثال :

• شرکت IBM بازبینی 7053 ساعته
۲۰۰ هزار خط کد و یافت ۳۱۱۲
اشکال و در کل هزینه ۲۸۲ هزار دلار
برای پیشگیری

• اگر پیشگیری وجود نداشت و در
شرایطی خوب فقط ۲۰۰ اشکال و
۲۵۰۰۰ دلار برای هر خطا سبب
هزینه ۵ میلیون دلاری برای اصلاح
خطا

بهره برداری

حرکت کیفی

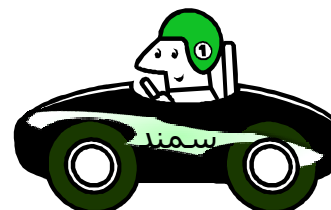
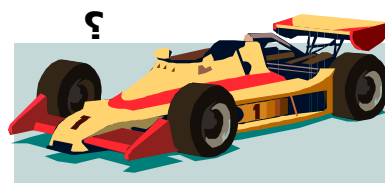
امروزه مدیران مسئول کمپانی های سراسر جهان صنعتی فهمیده اند که تولید یا کیفیت بالا منجر به صرفه جویی در هزینه ها و محصول پایانی بهتری شود. ژاپنی ها با مبنای قراردادن عقاید آقای **دمینگ** روشهای **سیستماتیک** را برای **حذف عوامل ریشه ای** اشکالات در محصولات توسعه دادند. کار آنها به جهان غرب منتقل شد و نامهایی چون مدیریت کیفیت یکپارچه و کامل (TQM) به آن داده شد اگر چه کمپانی ها و نویسندگان مختلف اصطلاحات متفاوتی در این زمینه به کار می برند اما هر برنامه TQM معمولاً با یک توالی چهار مرحله ای اصلی به عنوان زیر بنا دارد: **(مراحل TQM)**

مرحله اول: کایزن گفته میشود. سیستمی با روند پیشرفت دائمی دارد. هدف آن به وجود آوردن فرایندی قابل مشاهده قابل تکرار و قابل سنجش و اندازه گیری است.

مرحله دوم: که پس از به نتیجه رسیدن مرحله کایزن آغاز می گردد. اتاریما هینشیتسو گفته میشود. این مرحله به بررسی عوامل پنهانی موثر بر فرآیند می پردازد و تلاش می کند تاثیر آنها بر روند کار را خوش بینانه نماید.

• **مرحله سوم** که کانسئی (معنی: حواس پنج گانه) گفته میشود به جای فرآیند بر مصرف کننده محصول تمرکز میکند. در اساس با بررسی چگونگی کاربرد محصول از سوی مصرف کننده کانسئی به بهبود بخشیدن خود محصول می پردازد و بطور بالقوه نیز فرایندی را که به وجود آورده است تقویت میکند.

مرحله چهارم: میریوکوتکی هینشیتسو. این مرحله، مرحله ای تجارت گراست که به دنبال یافتن فرصت در زمینه های مشابه است. و با مشاهده کاربرد کالا در بازار قابل شناسایی اند. در جهان نرم افزار میریوکوتکی هینشیتسو میتواند تلاشی برای آشکار کردن تولیدات و کاربردهای نو و سود آور در نظر گرفته شود که پیامد سیستم های مبتنی بر کامپیوتر موجود میباشد. (شناسایی محصولات پرقایده) در بیشتر شرکت ها توجه به کایزن باید در اولویت باشد. (پیش نیاز سایر مراحل)



تضمین کیفیت نرم افزار

تعریف کیفیت نرم افزاری عبارت است: ۱) تطابق با نیازهای عملکردی و کارکردی که به وضوح بیان شده اند. ۲) استانداردهای توسعه ای که بی هیچ ابهامی ثبت گردیده اند. و ۳ ویژگی های تردیدناپذیری که وجود آنها در هر نرم افزار حرفه ای توسعه یافته ای انتظار می رود. از تعریف فوق برای تاکید بر سه نکته مهم استفاده میشود:

- ۱) نیازمندیهای نرم افزاری مبنای اندازه گیری کیفیت است. فقدان هم سویی با نیازمندیها فقدان کیفیت است.
- ۲) استاندارد های خاصی یک سری معیارهای توسعه را تعریف می کنند که مسیر مهندسی نرم افزار را مشخص میکند. اگر این معیار ها دنبال نگردند تقریباً به طور حتم فقدان کیفیت رخ خواهد نمود
- ۳) یک سری نیازمندیهای ضمنی وجود دارد که اغلب به صراحت بیان نشده اند. اگر نرم افزار با نیازمندیهای اصلی اش سازگار باشد اما نیازمندیهای ضمنی خود را برآورده نسازد کیفیت نرم افزار مورد شک قرار می گیرد.

موضوعات زمینه

پیشینه تضمین کیفیت در توسعه نرم افزار موازی با پیشینه کیفیت در ساخت سخت افزار است. در خلال **اولین** روزهای به وجود آمدن کامپیوتر **کیفیت** تنها **وظیفه برنامه نویس** بود. استاندارد های تضمین کیفیت برای نرم افزار در دهه ی ۷۰ برای نخستین بار در قرارداد های نظامی توسعه نرم افزار به وجود آمد و سپس به سرعت به توسعه نرم افزار در جهان تجارت گسترش یافت. با بسط تعریف ارائه شده قبلی کیفیت نرم افزار **"الگوی نظام مند و برنامه ریزی شده فعالیت هاست. که برای تضمین کیفیت نرم افزار الزامی اند.** شاید به بهترین نحو بتوان عرصه مسئولیت تضمین کیفیت را با شعار سازندگان اتومبیلی که زمانی مورد توجه همگان بود مشخص کرد: **"حرف نخست را کیفیت می زند"** پس گروههای مختلف چون مهندسين نرم افزار ، مدیران پروژه، مشتری، فروشندگان و افراد SQA در تضمین کیفیت نقش دارند.

گروه SQA به عنوان نماینده دائم مشتری عمل می نمایند یعنی اجرا کنندگان SQA باست از دید مشتری به نرم افزار نگاه کنند



فعالیت های تضمین کیفیت نرم افزار (SQA)

تضمین کیفیت نرم افزار از انواع گوناگونی از وظائف تشکیل یافته است که به دو گروه متفاوت مربوط می باشد.

۱. **مهندسان نرم افزار** که کار فنی انجام می دهند

۲. **یک گروه SQA** که مسئولیت برنامه ریزی تضمین کیفیت نظارت ثبت و ضبط تجزیه و تحلیل و گزارش را بر عهده دارند.

مهندسان نرم افزار با بکارگیری روشهای فنی مشخص و معیارها؛ اجرای آزمونهای نرم افزاری خوب کیفیت را مورد توجه قرار دارند فعالیت گروه SQA، هدایت تیم نرم افزار برای دستیابی به محصول نهایی با کیفیت بالا می باشد فعالتهای SQA توسط گروه مستقلی انجام می شود که

(۱) **طرح SQA** برای پروژه آماده می کند. طرح در مدت برنامه ریزی پروژه به وجود می آید و توسط تمام گروه های علاقه مند **بررسی** می شود. طرح فعالیت های تضمین کیفیت توسط تیم مهندسی نرم افزار اجرا می شود و گروه SQA را مشخص می کند. طرح به تشخیص موارد ذیل می پردازد:

- ارزشیابی که باید انجام گیرد.
- استانداردهایی که درمورد پروژه قابل اعمال اند.
- مستنداتی که باید توسط گروه SQA به وجود آیند.
- مقدار باز خوردی که به تیم پروژه نرم افزاری بازگردانده شده است.

(۲) **در توصیف فرآیند پروژه نرم افزاری شرکت می کند:** توصیف فرآیند برای سازگاری با استانداردهای داخلی نرم افزار و استانداردهای شدیدا الزامی خارجی (تحمیلی)

(۳) **بررسی تطابق فعالتهای مهندسی نرم افزار و فرآیند:** انحرافها مستند سازی می شوند و پی گیری و بازبینی برای اصلاح

(۴) **بررسی تطابق محصول کاری نرم افزار با محصول کاری تعیین شده:** انحرافها مستند سازی می شوند و پی گیری و بازبینی برای اصلاح

(۵) **حصول اطمینان از مستندسازی انحرافات و طبق یک روال مستند اداره می گردند**

(۶) **هرنوع ناهماهنگی ثبت شده و به مدیر ارشد (عالی) گزارش می دهد.**



باز بینی های نرم افزار

۱. بررسی های نرم افزار "فیلتری" برای فرایند مهندسی نرم افزار می باشند. یعنی در خلال توسعه نرم افزار در نقاط مختلف بررسیها اعمال میشوند و از آنها برای **شناسایی خطاها و نقص ها** و سپس **برطرف کردن آنها** استفاده می شود. بررسی های نرم افزاری به منظور "پاک کردن" فعالیت های مهندسی نرم افزاری که ما آنها را تجزیه و تحلیل طراحی و برنامه نویسی می گوئیم اعمال میشوند. و برای کارفنی مثل **پاک کن برای مداد** هستند.

۲. معمولا افراد در پیدا کردن خطاهای خود خوب عمل می کنند ولی دسته ای از خطاها خیلی آسانتر از دید کننده کار پنهان می مانند یک بررسی راهی است برای استفاده از مسیرگروهی از مردم به منظور:

(۱) نشان دادن اصلاحات لازم در تولید توسط یک فرد تنها یا یک گروه

(۲) تعیین آن بخش های از تولید که یا علاقه ای به اصلاح آنها وجود ندارد یا اصولا نیازی به اصلاح شدن ندارند

(۳) دست یابی به کار فنی یک دست تر یا دست کم قابل پیش بینی تر به منظور بالا بردن قابلیت

مدیریت کار که در نتیجه آن می توان بدون انجام بررسی به کیفیت دست یافت.

موثرترین روش فیلتر ، روش بررسی فنی رسمی (وارسی یا بازرسی) (FTR) و باعث ایجاد کیفیت می شود.



تأثیر عیوب بر هزینه نرم افزاری

در واژه نامه ها IEEE: **نقص** عبارت است از یک ناهنجاری در محصول و **عیب** هم تعریف شده است از اشکالی در دستگاه یا یک مرحله فرایندیا تعریف داده اشتباه

در نرم افزار این دو واژه مترادف و هم معنی هستند و به وجود اشکال کیفی آشکار شده بعد از تحویل محصول اشاره دارند. لازم به یادآوری است. واژه **خطا** ، قبلا برای اشاره به مشکلی در کیفیت که قبل از تحویل آشکار شده استفاده شد.

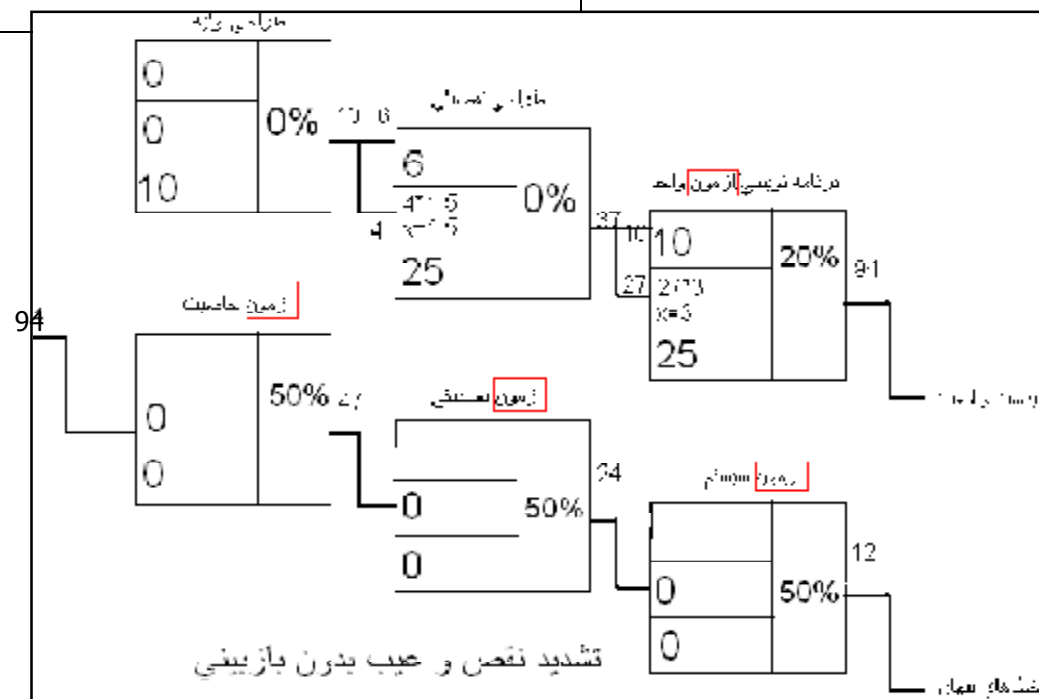
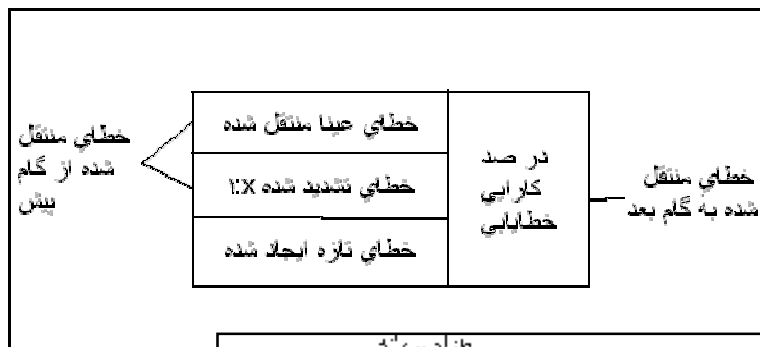
اولین هدف بررسی های فنی رسمی (FTR) یافتن خطاها در جریان فرایند است به طوری که آنها بعد از عرضه نرم افزار تبدیل به نقص نگردد. مزیت اشکار بررسی های فنی رسمی کشف زود هنگام خطاست به طوری که آنها نتوانند به مرحله بعدی در فرایند نرم افزار نشر یابند. تعدادی از مطالعات صنعتی نشان می دهد که فعالیت های طراحی بین ۵۰ تا ۶۵ درصد خطاها را در خلال فرایند نرم افزاری ایجاد می کنند. با این حال نشان داده شده است که فنون بررسی رسمی تا ۷۵ درصد در اشکار کردن نقایص طراحی موثر می باشد. با شناسایی و رفع درصد بالایی از این خطاها فرایند بررسی ها به طور عمده ای هزینه مراحل بعدی توسعه و تضمین را کاهش داده است.

تشدید نقص و برطرف کردن آن

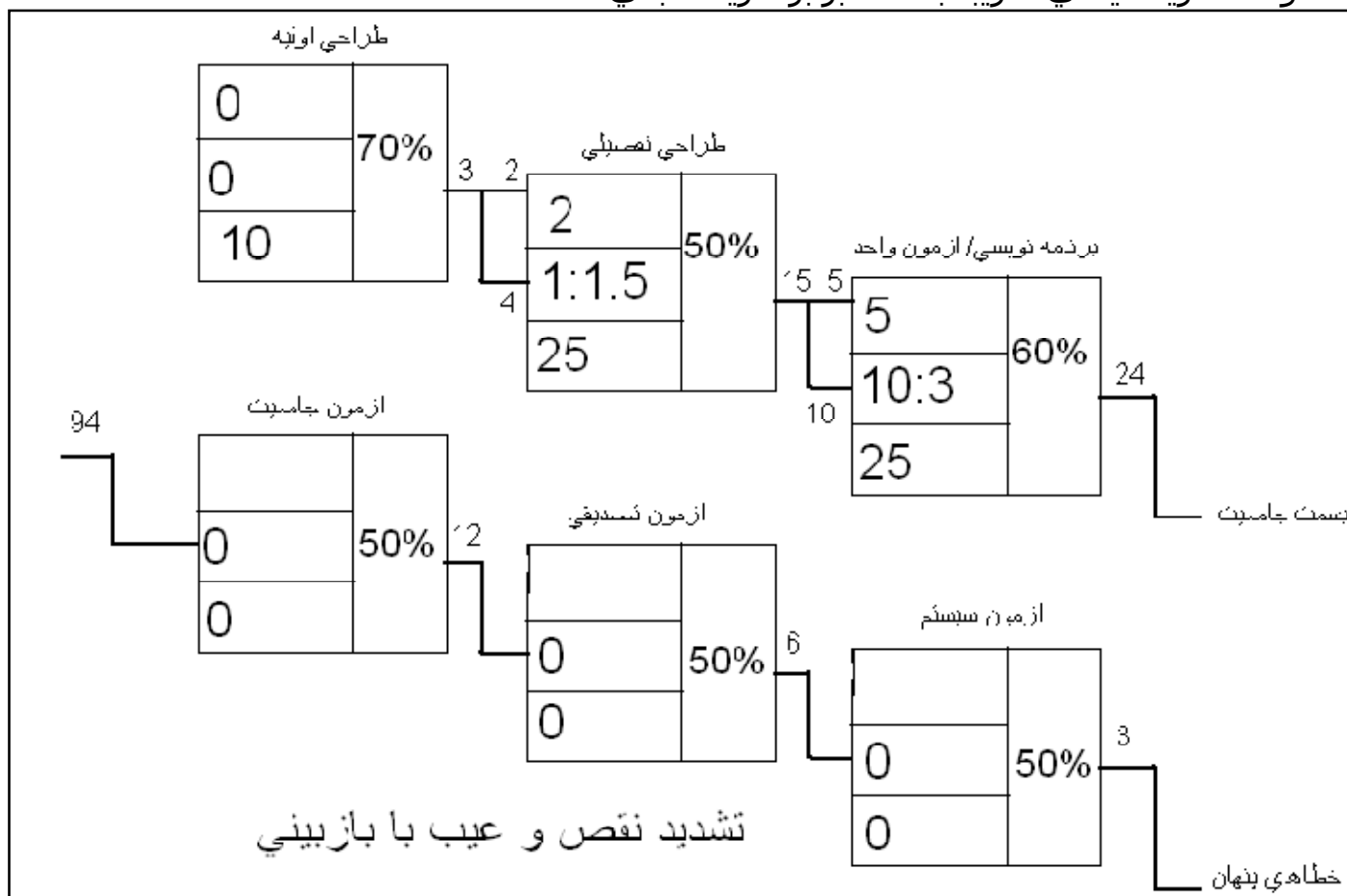
يك مدل توسعه نقص براي نمایش تولید و رد يابي خطاها در جریان طراحی اولیه ،طراحی تفصیلی و جزئیات و مراحل برنامه نویسی فرایند مهندسی نرم افزار قابل کاربرد است. این مدل به صورت نمودار در تصویر زیر نشان داده شده است.

هر جعبه يکي از مراحل توسعه نرم افزار را نشان مي دهد. در جریان هر مرحله خطاها ممکن است سهوا تولید شوند. بررسی ها ممکن است نتوانند خطاهای جدید و یا خطاهای قبلی را آشکار کنند و در نتیجه تعدادی خطا را دوباره فعال کند. در بعضی موارد خطاهای باقی مانده از مراحل قبلی به وسیله کار کنونی **گسترش می یابند**. تقسیمات فرعی جعبه ها نمایانگر هر يك از این ویژگی ها و درصد کارایی خطاهای ردیابی شده مي باشد که کارکردی از اجرای کامل بررسی ها می باشد.

• تصویر زیر يك مثال فرضي از توسعه و تشدید نقص در يك فرایند توسعه نرم افزاری را نشان مي دهد که در آن هیچ گونه بررسی انجام نشده است. با توجه به تصویر فرض می شود که هر يك از مراحل آزمون برای آشکار سازی و تصحیح ۵۰ درصد از همه خطاهای کنونی بدون تولید خطاهای جدید به کار میرود. قبل از آغاز مراحل آزمون ده نقص طراحی اولیه به ۹۴ خطا گسترش یافته و تشدید شده اند. دوازده خطای پنهان نیز وارد محیط کار شده اند.



تصویر زیر نیز شرایطی مشابه را مورد ملاحظه قرار می دهد با این استثنا که بررسی طراحی و برنامه به عنوان بخشی از هر مرحله انجام گرفته است. در این مورد ده خطای طراحی اولیه قبل از آغاز آزمون ها به ۲۴ خطا افزایش یافته اند-تنها سه خطای پنهان وجود دارد. با به یاد آوردن هزینه های نسبی مربوط به کشف و تصحیح خطاها ارزش کلی قابل بیان خواهد بود. تعداد خطاهای آشکار شده در هر یک از مراحل مورد اشاره تصاویر با هزینه برطرف کردن هر خطا افزایش یافته است. با به کار بردن این داده ها هزینه کلی توسعه و نگهداری در جریان انجام بررسی ها برابر است با ۷۸۳ واحد هزینه. اگر هیچ گونه بررسی ای انجام نگرفته باشد هزینه کلی برابر خواهد بود با ۲۱۷۷ واحد هزینه-یعنی تقریباً با سه برابر هزینه قبلی.



بازبینی های فنی - رسمی

یک بررسی فنی _ رسمی (FTR) یک فعالیت تضمین کیفیت است که توسط مهندسان نرم افزار (و دیگران) اجرا می شود. FTR درواقع گروهی از بررسی هاست که شامل جستجوها، بازرسی ها، بررسی های مخفی و دیگر ارزیابی های فنی گروه های کوچک نرم افزاری است. اهداف آن عبارتند از:

۱. آشکار کردن خطاهای موجود در کارکرد ، یا اجرای هر نوع ارائه نرم افزار.
 ۲. بررسی این نکته که نرم افزار تحت بررسی با نیازمندی هایش هم خوانی دارد.
 ۳. اطمینان از اینکه نرم افزار بر اساس استانداردهای از پیش تعریف شده ارائه شده است.
 ۴. دستیابی به نرم افزاری که به شیوه ای همگن (یکنواخت) تولید شده باشد.
 ۵. افزایش قابلیت اداره پروژه ها.
- علاوه براین FTR به عنوان یک زمینه آموزشی به کار می رود، هم چنین موجب تقویت پیشرفت و انسجام کار می شود.

نشست بازبینی

هر FTR به صورت يك جلسه هدایت می شود، صرف نظراز قالبی که برای FTR انتخاب شده است ، هرگردد هم آیی بررسی باید به وسیله قیود ذیل به پیش رود.

- ❖ معمولاً سه تا پنج نفر در این بررسی دخالت داشته باشند
- ❖ آمادگی قبلی باید وجود داشته باشد.
- ❖ مدت زمان گردهم آیی بررسی باید کمتر از دو ساعت باشد.

FTR بر روی کل نرم افزار تاکید ندارد بلکه **برای هر محصول میانی به طور مجزا برپا می گردد**

تاکید FTR بر **محصول کاری است** و با اتمام ساخت يك محصول کاری از طریق مسئول پروژه از مسئول تیم مرور درخواست جلسه می شود . این جلسه با حضور ^۱سازنده محصول کاری، ^۲مرورگرها و ^۳مسئول بررسی شروع می شود که یکی از مرورگرها نقش منشی و ثبت کننده را به عهده دارد

در انتهای جلسه يك گزارش نتیجه گیری نوشته می شود .

قبول محصول بدون اصلاح ، رد کردن محصول ، قبول محصول با در نظر گرفتن یکسری نکات

گزارش بازیابی ها و ثبت موضوعات

منشی و ثبت کننده تمام موضوعات مطرح شده در جلسه مرور را ثبت می کند و يك لیست از موضوعات مطرح شده و يك گزارش ایجاد می شود که این گزارش مختصری از بررسی فنی رسمی امور تهیه می گردد. این گزارش به سه سوال پاسخ می دهد:

۱. چه چیزی بررسی شده است؟

۲. چه کسی آن را بررسی کرده است؟

۳. نتایج یافته ها چه چیزی بودند؟

لیست موضوعات بررسی شده نیز به دو منظور به کار می رود:

۱. شناسایی زمینه های مشکل دار محصول

۲. به عنوان چک لیست عملکردها برای راهنمایی تولید کننده

رهنمودهای بازیابی

يك بررسی FTR بایست درست و کنترل شده باشد . استفاده از رهنمودهای زیر برای انجام FTR توصیه می شود:

۱. محصول را بررسی کنید نه تولیدکننده را: این کار در فضای آرام و سازنده باشد نه فضای خشک و بی روح

۲. یک دستور کار تنظیم و آن را حفظ کنید: ایجاد دستور کار برای زمانبندی و جلوگیری از انحراف

۳. منازعه و جدل متقابل را محدود کنید.

۴. زمینه های دارای مشکل را اعلام دارید، اما سعی نکنید تمام مشکلات اعلام شده را حل کنید.

۵. یادداشت های کتبی تهیه کنید: ثبتی ها در معرض دید همه باشند

۶. تعداد شرکت کنندگان را محدود کنید و برآمدگی قبلی تاکید کنید.: جلسه ۱۴ نفری نشانه برتری بر جلسه ۴ نفری نیست

۷. لیستی برای هر محصول که احتمال بررسی آن می رود تهیه کنید.

۸. منابع و زمان لازم به FTR تخصیص دهید.

۹. آموزش های هدفمندی برای همه بررسی کنندگان (مرورگرها) فراهم کنید.

۱۰. بررسی های اولیه ی خود را بررسی کنید. (خلاصه سازی بررسی ها)



رهیافتهای (روش) رسمی برای تضمین کیفیت نرم افزار (SQA)

کیفیت نرم افزار برعهده همه افراد است که در مراحل تحلیل و برنامه سازی و آزمون همان قدر باید مورد توجه باشد که در باز بینی های رسمی فنی ، در راهبرد های تلاش چندگانه آزمون و کنترل بهتر محصولات کاری و تغییر آنها و دریافت استانداردهای مهندسی نرم افزار کیفیت می تواند به طیف گسترده ای از عوامل کیفی و اندازه های غیرمستقیم تعبیر شود که شاخص ها و متریک های متعدد آن را نشان می دهند . یک بخش کوچک اما ویژه از جامعه نرم افزار معتقدند روشی رسمی برای تضمین کیفیت نرم افزار مورد نیاز است آنها معتقدند که یک برنامه کامپیوتری یک شی ریاضی محسوب می شود و یک **نحو و معنای ثابتی** میتواند در هر زبان برنامه سازی تعریف شود که در این صورت رهیافت ثابتی نیز برای مشخصه های نیازمندی های نرم افزاری در پیش رو خواهد بود.

تضمین کیفیت نرم افزار به صورت آماری (SSPI)

تضمین کیفیت آماری منعکس کننده يك روند رو به رشد در صنعت برای **کمی نمودن کیفیت** است در مورد نرم افزار تضمین کیفیت آماری متضمن مراحل زیر است:

۱. **اطلاعات درباره نقص های نرم افزار جمع آوری و دسته بندی می شود.**
 ۲. **تلاشی برای ردیابی علت زیربنایی (اصلی) هر نقص انجام می گیرد.**
 ۳. **اصول پارتو به کار بسته می شود.** (۸۰ درصد مشکلات در ۲۰ درصد علل قرار دارند) تا ۲۰ درصد **شناسایی** شود
 ۴. **در حین شناسایی علل حیاتی (مرحله قبل) حرکت به سمت اصلاح مسائلی که ریشه نواقص بوده اند انجام می گیرد**
- به منظور کاربرد SQA آماری جدول زیر طراحی شده است. این جدول نشان می دهد که MCC و IES و EDR به عنوان دلایل اصلی انتخاب خواهند شد در حالیکه ۵۳ درصد تمام خطاها هستند. بعد از اینکه دلایل اساسی تعیین گردیدند سازمان مهندسی نرم افزاری ممکن است برای بهبود کیفیت مشتری و مشخصات تعیین شده، به جای اجرای فنون مشخصات کاربرد تسهیل شده بپردازد.



نام درس: مهندسی نرم افزار ۱

تهیه کننده: ایمان مختاری
مدرس دانشگاههای پیام نور شهر کرد- فارسان- فرخ شهر
[Http://www.Snu.ir](http://www.Snu.ir)

شماره صفحه : ۱۲۲

	E کل			S مهم		M متوسط		T جزئی	
	نوع خطا	تعداد	درصد	تعداد	درصد	تعداد	درصد	تعداد	درصد
• incomplete or erroneous specifications (IES)	IES	205	22%	34	27%	68	18%	103	24%
• misinterpretation of customer communication (MCC)	MCC	156	17%	12	9%	68	18%	76	17%
• intentional deviation from specifications (IDS)	IDS	48	5%	1	9%	24	6%	23	5%
• violation of programming standards (VPS)	VPS	25	3%	0	1%	15	4%	10	2%
• error in data representation (EDR)	EDR	130	14%	26	20%	68	18%	36	2%
• inconsistent component interface (ICI)	ICI	58	6%	9	7%	18	5%	31	7%
• error in design logic (EDL)	EDL	45	5%	14	11%	12	3%	19	4%
• incomplete or erroneous testing (IET)	IET	95	10%	12	9%	35	9%	48	11%
• inaccurate or incomplete documentation (IID)	IID	36	4%	2	2%	20	5%	14	3%
• error in programming language translation (PLT)	PLT	60	6%	15	12%	19	5%	26	6%
• inconsistent human/computer interface (HCI)	HCI	28	3%	3	2%	17	4%	8	2%
• miscellaneous (MIS)	MCI	56	6%	0	0%	15	4%	41	9%
	TOTAL	942	100%	128	100%	379	100%	435	100%

نکته : فعالیت اصلاحی در وهله نخست **برعلل اساسی** متمرکز است، با تصحیح علل اساسی، مسایل جدیدی برای تصحیح شدن پیدا می شود. فنون تضمین کیفیت آماری برای نرم افزار، نشان دهنده فراهم آوردن بهبود کیفیت آماری است. در کنار جمع آوری اطلاعات درباره نقایص کار سازندگان نرم افزار می تواند یک شاخص خطا (EI) برای هر یک از مراحل اصلی فرآیند محاسبه کنند. بعد از تجزیه و تحلیل، طراحی، برنامه نویسی، آزمودن و عرضه، اطلاعات زیر جمع آوری شده اند.

Ei : تعداد کل خطاهایی که در خلال مرحله i ام روند مهندسی نرم افزار

Si : تعداد خطاهای اساسی

Mi : تعداد خطاهای میانه

Ti : تعداد خطاهای کوچک

PS : اندازه محصول (از لحاظ تعداد خطوط، ریال، مستندات و...) در مرحله i ام

Ws, Wm, Wt : ضریب ارزش خطاهای اساسی - متوسط و جزئی

serious errors, moderate errors,

در هر مرحله از فرآیند نرم افزاری یک شاخص مرحله PI_i محاسبه می شود:

$$PI_i = W_s (S_i / E_i) + W_m (M_i / E_i) + W_t (T_i / E_i)$$

شاخص خطا E_i با محاسبه اثر انباشتی (تجمعی) PI_i محاسبه شده است:

$$EI = \sum (i * PI_i) / PS = (PI_1 + 2PI_2 + 3PI_3 + \dots + iPI_i) / PS$$

برای توسعه یک نشان کلی از بهبود به وجود آمده در کیفیت نرم افزاری می توان شاخص خطا را به همراه اطلاعات گردآوری شده در جدول قبل به کار برد. کاربرد SQA آماری وقاعده پارتو را در یک جمله واحد می توان خلاصه کرد: زمان خود را صرف تمرکز بر چیزهایی کنید که حائز اهمیت هستند، اما نخست مطمئن شوید که شما واقعا درک می کنید چه چیزی مهم است.

قابلیت اطمینان (اعتبار) نرم افزار

قابلیت اطمینان برای یک برنامه عنصر مهمی در کنترل کیفیت نرم افزار می باشد. اعتبار نرم افزار برخلاف بسیاری دیگر از فاکتورهای کیفیت با به کار بردن اطلاعات تاریخی و توسعه ای به طور مستقیم قابل اندازه گیری است و می توان آن را برآورد کرد. در اصطلاحات آمار اعتبار نرم افزاری به صورت "احتمال عملکرد بدون خرابی یک برنامه کامپیوتری در یک محیط خاص و در یک زمان مشخص".

به عنوان نمونه : برنامه X برآورد شده است و دارای اعتبار ۰.۹۶ است در طول ۸ ساعت فرایند سپری شده. مفهوم : اگر برنامه X برای ۱۰۰ بار به اجرا در آمدن طراحی شده باشد و نیاز به ۸ ساعت زمان فرایند سپری شده داشته باشد (زمان اجرا) احتمال دارد که ۹۶ بار از ۱۰۰ بار مورد انتظار را به طور صحیح عمل نماید. **خرابی چیست؟** دریافت هر بحثی درباره کیفیت و اعتبار نرم افزاری، خرابی یا ناسازگاری با نیازمندی های نرم افزار. خرابی دارای درجه بندی می باشد. دو درجه بندی خرابی عبارتند از :

۱. **اذیت کننده:** در چند ثانیه تصحیح می شود.
۲. **مصیبت بار:** نیاز به هفته ها یا حتی ماهها برای تصحیح شدن دارد.



Measures of Reliability and Availability

اندازه گیری قابلیت اطمینان (اعتبار) و دسترسی

اندازه گیری قابلیت اطمینان (اعتبار)

کارهای اولیه بر روی اعتبار نرم افزاری تلاش داشتند تا ریاضیات تئوری اعتبار سخت افزاری را برای پیش بینی اعتبار نرم افزاری به کار برند.

خرابی ها ناشی از	سخت افزار	نرم افزار
کهنگی و فرسودگی	قسمت اعظم	نقش کمرنگ
مشکلات طراحی یا پیاده سازی	نقش کم رنگ	نقش اساسی

اگر یک سیستم کامپیوتری را مورد مطالعه قرار دهیم اندازه گیری ساده اعتبار و قابلیت اطمینان عبارت خواهد بود از:

$$MTBF = MTTF + MTTR$$

MTBF (**meantime-between-failure**): متوسط زمان بین دو شکست

MTTF (**mean-time-to-failure**): فاصله زمانی بروز یک اشکال

MTTR (**mean-time-to-repair**): فاصله زمانی برطرف کردن آن اشکال

بسیاری محققان معتقدند که MTBF یک اندازه گیری بسیار مفیدتر از خطاها در KLOC یا خطاها در FP است. آن ها این بیان ساده را مطرح می کنند که یک مصرف کننده نهایی با خرابی ها سروکار دارد نه محاسبه خطای کلی. از آنجایی که هر خطایی که در یک برنامه موجود است دارای نرخ خرابی یکسانی نمی باشد، محاسبه خطای کلی نشان اندکی از اعتبار سیستم به دست می دهد.

اندازه گیری میزان دسترسی

قابلیت دسترسی: قابلیت دسترسی نرم افزار احتمال عملکرد یک برنامه بر طبق نیازمندیهای خاص در یک نقطه زمانی است و به صورت $Availability = [MTTF / (MTTF + MTTR)] * 100\%$ تعریف می شود:

در میزان اعتبار MTBF زمانهای MTTF و MTTR به یک میزان موثرند. میزان قابلیت دسترسی تا اندازه بیشتری به MTTR حساس است، که میزان غیرمستقیمی از قابلیت نگهداری نرم افزار است.



ایمنی نرم افزار

ایمنی نرم افزار یک فعالیت تضمین کیفیت نرم افزار است که بر شناسایی و ارزیابی خطرات بالقوه ای ممکن است بر روی نرم افزار منفی داشته باشند و موجب خرابی یک سیستم کلی گردند تمرکز دارد یک فرایند مدل سازی و تجزیه و تحلیل به عنوان بخشی از ایمنی نرم افزار اجرا شده است.

قدم اول : شناسایی خطرات . (مثلا ورود حرف به جای عدد در فیلد تاریخ)

قدم دوم : به کارگرفتن فنون تجزیه و تحلیل برای تعیین شدت یا احتمال اتفاق افتادن خطرات .

قدم سوم : تعیین نیازمندی های مربوط به ایمنی نرم افزار .

تفاوت اعتبار نرم افزار (شکستها) و ایمنی نرم افزار (خطرات)

اگرچه اعتبار نرم افزار و ایمنی نرم افزار ارتباط نزدیکی باهم دارند، درک تفاوت ظریف بین آنها دارای اهمیت است. اعتبار و قابلیت اطمینان نرم افزار برای تعیین احتمال این که یک خرابی نرم افزاری اتفاق بیفتد از تجزیه و تحلیل آماری استفاده می کند، با این وجود رخ دادن یک خرابی الزاما منجر به یک خطری اتفاق سوء نمی شود.

ایمنی نرم افزار به آزمون راههایی می پردازد که از طریق آنها خرابی ها منجر به شرایطی می شوند که ممکن است به یک اتفاق سوء یا نجامد، این بدان معنی است که خرابی ها در حلال در نظر گرفته نشده اند، بلکه دریافت سیستمهای کامپیوتری ارزشیابی شده اند.

مصونیت نرم افزار در برابر اشتباه (نرم افزار ضد خطا)

روش پوکا یوک (ضد اشتباه) یک تکنیک تضمین کیفیت که نتیجه آن پیشگیری یا پیش گیری و تصحیح زود هنگام خطاها در روند ساخت است از ابزارهای پوکا-یوک (مصونیت از اشتباه) استفاده می کند. مکانیزمهایی که منجر به (۱) جلوگیری از مسائل کیفی بالقوه قبل از رخ دادن آنها شد (مثلا ماشین در صورت نبستن تمام درها حرکت نکند) یا (۲) تشخیص سریع مسائل کیفی در صورتی که وارد کار شده باشند، می شد (مثلا در صورت کمبود بنزین آلامر داده شود....). یک ابزار پوکا-یوک کارآر اداری یک سری ویژگیهایی است:

۱. ساده و ارزان باشد.

۲. بخشی از فرآیند مهندسی است است.

۳. در نزدیکی وظیفه ای از فرآیند که اشتباهات رخ می دهند. که باعث فیدبک سریعی می شود و صرفه جویی در زمان کشف مشکل

استانداردهای کیفیت ایزو ۹۰۰۰

ایزو ۹۰۰۰ ، عناصر تضمین کیفیت را به صورت عمومی توصیف می کند که می توانند برای هر شغلی و با هر نوع محصولات و سرویسهایی که ارائه می دهند به کار گرفته شود

این استاندارد که توسط بیش از ۱۲۰ کشور پذیرفته شده است به طور مستمر بر اهمیت آن به عنوان ابزاری که به وسیله آن مشتریان می توانند توانایی سازندگان نرم افزار را مورد قضاوت قرار دهند افزوده شده است. یکی از مشکلات (؟؟؟) مربوط به سری استانداردهای Iso9001 مربوط نبودن آنها به صنعت خاصی است. این استاندارد با اصطلاح های عمومی بیان گردیده و به وسیله تولید کنندگان محصولات مختلفی قابل تعبیر است.



رهیافت ایزو جهت سیستم های تضمین کیفیت

برای صنعت نرم افزار استانداردهای مربوط عبارتند از :

- **Iso9001** مدلی برای تضمین کیفیت در طراحی ، توسعه، تولید، نصب و خدمات دهی است.
 - **Iso9000** توصیه هایی است برای به کارگیری Iso9001 در کار توسعه و در عرضه و نگهداری نرم افزار و سند خاصی است که Iso9001 را برای تولیدکننده نرم افزار معنی می کند.
 - **Iso9004** مدیریت کیفیت و عناصر سیستم کیفیت. و حمایت از مصرف کننده
- ایزو ۹۰۰۱ در نرم افزار شامل ۲۰ نیازی است که باید برای یک سیستم تضمین کیفیت موثر وجود داشته باشد این نیازمندیها در زیر ۲۰ عنوان گروه بندی شده است: مسئولیت مدیریت، تجهیزات آزمون و اندازه گیری بازرسی، سیستم کیفیت، بازرسی وضعیت آزمون، بررسی قرارداد، اقدام اصلاحی، کنترل طرح ، کنترل بر محصول ناسازگار، کنترل سند، جابجایی، انبار، بسته بندی و عرضه، خرید، ثبت کیفیت، محصول عرضه شده خریدار، بازرسی درونی کیفیت، قابلیت پیگیری و تشخیص تولید، آموزش، کنترل مرحله ای، سرویس کردن، بازرسی و آزمون ، فنون آماری.

استاندارد ایزو ۹۰۰۱

به منظور حفظ مطابقت محصول با نیازمندیهای ساخت آن، تولیدکننده باید به کنترل، تعیین و حفظ بازرسی، اندازه گیری و فراهم کردن وسایل آزمون محصول بپردازد.

اولین چیزی که باید بدان توجه کرد **عمومیت** آن است، آن را می توان در مورد تولید کننده هر محصولی به کاربرد. دومین چیزی که باید به آن توجه کرد **دشواری تعبیر پاراگراف** است. (این پاراگراف به روشنی بر مراحل مهندسی استاندارد و همزمان به کیفیت ابزارها و تجهیزات تاکید دارد)



طرح تضمین کیفیت نرم افزار SQA

این طرح که به وسیله گروه SQA به وجود آمده راهی را برای ایجاد تضمین کیفیت نرم افزار فراهم می کند و به عنوان الگویی برای فعالیتهای SQA ارائه می شوند که برای هر پروژه نرم افزاری برقرار گردیده اند. مثلاً ISO9000 طرحی است برای ISO9001

استانداردی به وسیله IEEE برای **طرح های SQA** پیشنهاد شده است.

۱- بخش مدیریت : طرح جایگاه SQA را در ساختار سازمانی ، کارهای SQA و فعالیتهای آنها و جایگاهشان در فرآیند نرم افزاری ، و نقشهای سازمانی و مسئولیتهای مربوط به کیفیت محصول را توصیف می کند.

۲- بخش مستندسازی (رجوع) : به هریک از محصولات کاری که به عنوان بخشی از فرآیند نرم افزاری تولید شده اند به توصیف آن ها می پردازد. این موارد عبارتند از :

- اسناد پروژه (مثلاً، طرح پروژه)
- مدلها (مثلاً، ERD ها ، سلسله مراتب طبقه بندی)
- اسناد فنی (مثلاً، مشخصات ، طرح های آزمون)
- اسناد کاربرد (مثلاً، فایل های کمک و راهنمایی)

علاوه بر این مجموعه حداقل محصولات کاری را معرفی می کند.

۳- بخش بررسی ها و واریسی های طرح : بررسی ها و واریسی هایی را که توسط تیم مهندسی نرم افزاری، گروه SQA و خریدار باید به عمل آیند مشخص می کند .

۴- بخش آزمون : این بخش به روال طرح آزمون نرم افزار نظر دارد. همچنین نیازمندیهای نگهداری نتایج ثبت شده آزمونها را مشخص می کند، گزارش مشکل و اقدام اصلاحی، مراحل گزارش، پیگیری و برطرف کردن اشکالات و نقایص را مشخص می کند، و مسئولیتهای سازمانی در برابر این فعالیتهای مشخص می کند.

قسمتهای دیگر طرح SQA ابزارها و روشهای تقویت فعالیتهای SQA را مشخص می کند.

خلاصه فصل هشتم (تضمین کیفیت)

تضمین کیفیت نرم افزاری یک "فعالیت چتری" است که در هریک از مراحل فرآیند نرم افزاری به کار بسته شده است. SQA در بردارنده مراحل به کار گیری موثر روشها، بررسی های فنی منظم، تکنیکها و راهبردهای آزمون، ابزارهای پوکا - یوک، روالهایی برای کنترل تضمین مطابقت با استانداردها، و سنجش مکانیزمهای گزارش کردن می باشد.

طبیعت پیچیده کیفیت نرم افزاری موجب پیچیدگی SQA شده است. یکی از خواص برنامه های کامپیوتری که به عنوان "مطابقت با نیازمندی های تعریف شده صریح و ضمنی" تعریف شده است، اما وقتی کیفیت نرم افزاری را بصورت عمودی تعریف کنیم بسیاری از محصولات مختلف، فاکتورهای پردازشی و متریک های مربوطه را در بر می گیرد. بررسی های نرم افزاری یکی از مهم ترین فعالیتهای SQA می باشند. بررسی ها به عنوان فیلترهایی در تمام فعالیت های مهندسی نرم افزاری عمل می کنند، که باعث برطرف شدن خطاها، زمانی که آنها برای یافتن و برطرف کردن گسترش زیادی نیافته اند، خواهند شد. بررسی فنی منظم یک گردهم آیی روش مند است که معلوم شده، برای مشخص کردن خطاها بسیار موثر است.

برای تضمین مناسب کیفیت نرم افزار، اطلاعاتی درباره فرآیند مهندسی نرم افزار باید گردآوری، ارزشیابی و پخش گردد. SQA آماری بر بهرکردن کیفیت محصول و خود فرآیند نرم افزار کمک می کند. مدل های اعتبار نرم افزار اندازه گیریهایی را بسط می دهد که برآورد و ارزیابی داده های گردآوری شده در مورد نقایص را در قالب نرخ های ناکامی موجود در طرح و پیش بینیهای اعتبار، ممکن می سازد. در پایان سخنان دان و آلن را بازگو می کنیم: "تضمین کیفیت نرم افزاری ترسیم مقررات مدیریتی و اصول طراحی تضمین کیفیت در فضای قابل کاربرد فنی و مدیریتی مهندسی نرم افزار است."



Software configuration management (SCM)

فصل ۹: مدیریت پیکربندی نرم افزار

□ مدیریت پیکربندی نرم افزار چیست؟

تغییرات در فرایند ساخت نرم افزار اجتناب ناپذیر است. تغییرات باعث **سردرگمی مهندسين و متخصصين** می شوند. مدیریت پیکربندی نرم افزار فعالیتی جامع است که در طول فرایند نرم افزاری اجرا می گردد (چون تغییرات در طول فرایند اتفاق می افتند) و دارای اهداف و مراحل زیر است:

- ۱- شناسایی تغییرات
- ۲- کسب اطمینان از حفظ کیفیت، با انجام تغییرات
- ۳- کنترل تغییرات
- ۴- گزارش تغییرات به افراد دستدرکار و علاقمند

□ چرا اهمیت دارد؟

اگر شما تغییرات را کنترل نکنید، تغییر شما را کنترل می کند وجود تغییرات باعث **سردرگمی** می شود و در روند انجام عملیات در یک پروژه موفق اختلال ایجاد می کنند

□ مراحل انجام آن کدامند؟

- ۱- شناسایی تغییرات ممکن در محصولات کاری
- ۲- مکانیزمهای کنترل تغییر
- ۳- حصول اطمینان از حفظ کیفیت حتی با انجام شدن تغییر
- ۴- گزارش تغییرات به کسانی که بایست بدانند

□ محصول کار چیست؟

طرح مدیریت پیکربندی نرم افزار (CSR)

□ تفاوت بین پشتیبانی از نرم افزار و مدیریت پیکربندی نرم افزار

پشتیبانی یعنی مجموعه ای از فعالیت های مهندسی نرم افزار که پس از تحویل نرم افزار به مشتری استفاده از آن به وقوع می پیوندد. مدیریت پیکربندی نرم افزار یعنی مجموعه ای از فعالیت های پیگیری و کنترل که زمانی شروع می شوند که یک پروژه مهندسی نرم افزاری آغاز می گردد و تنها زمانی پایان می یابند که نرم افزار از کار می افتد.

مدیریت پیکربندی نرم افزار (SCM):

خروجی فرایند نرم افزاری اطلاعاتیست که می توان آن را به سه گروه تقسیم کرد:

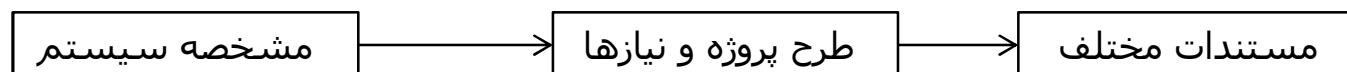
❖ برنامه های کامپیوتری

❖ مستندات که برنامه های کامپیوتری را شرح می دهد

❖ داده ها (در داخل یا خارج برنامه)

□ عوامل ایجاد سردرگمی

۱- در حال انجام فرایند تعداد آیت های پیکربندی نرم افزار (SCI) افزایش می یابد



تولید SCI جدید در اثر هر SCI باعث سردرگمی می شود

۲- متغیردیگری در فرایند نرم افزاری وجود دارد بنام "تغییر". تغییر می تواند در هر زمانی و به هر دلیلی روی دهد. در واقع اولین قانون مهندسی سیستم بیان می دارد که: مهم نیست که شما در کجای چرخه زندگی سیستم قرار دارید، سیستم تغییر خواهد کرد و تمایل به تغییر آن در تمام چرخه زندگی ادامه خواهد کرد.

□ چهار منبع تغییر اساسی وجود دارد:

(۱) تغییر در نیازهای محصول یا قوانین کار در اثر شرایط جدید کاری یا تغییرات بازار (نه تقصیر ما ، نه تقصیر مشتری)

(۲) در اثر نیازهای جدید مشتریان اصلاحات داده هایی که توسط سیستم های اطلاعاتی ایجاد می شوند، عملکردی که توسط محصولات عرضه می شوند، و یا خدماتی که توسط یک سیستم کامپیوتری ارائه می شوند (مقصر مشتری)

(۳) تغییرات در اولویت پروژه و یا ساختار تیم مهندسی نرم افزار در اثر سازماندهی دوباره و یا گسترش/کاهش تجارت (مقصر ما)

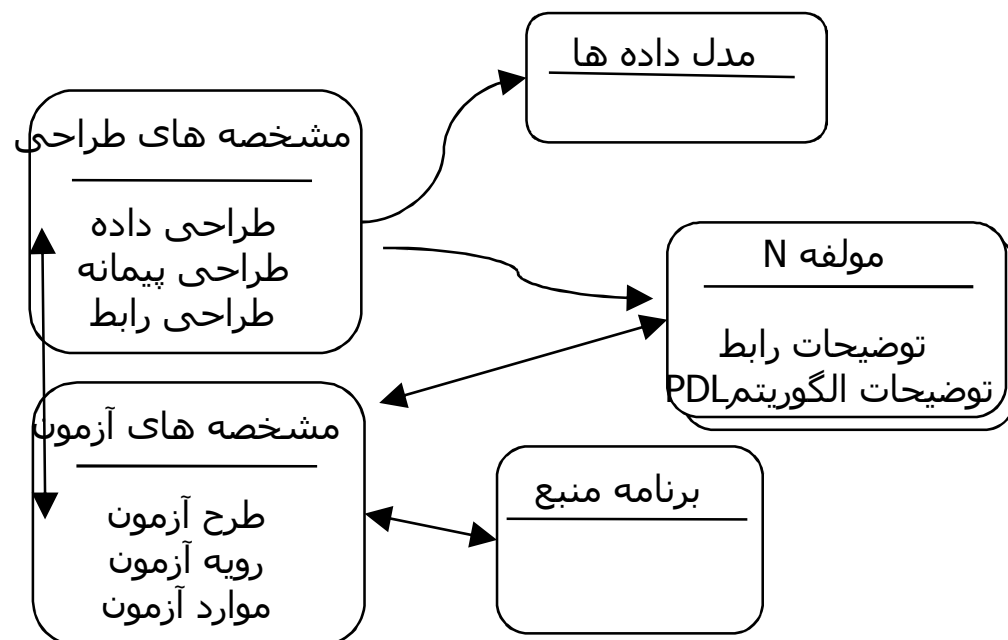
(۴) مشکلات مربوط به بودجه و یا برنامه ریزی که سبب تعریف دوباره سیستم و یا محصول می گردد. (مقصر مشتری)

مدیریت پیکربندی (وضعیت؟؟؟) نرم افزار یعنی مجموعه ای از فعالیت هایی که برای کنترل تغییر در طول چرخ زندگی نرم افزار کامپیوتری به وقوع می پیوندد.

اقلام پیکربندی نرم افزار (SCI)

اطلاعاتی که به عنوان بخشی از فرایند مهندسی نرم افزار به وجود می آیند. یک SCI عبارت است از یک **سند**، مجموعه کاملی از موارد **آزمون**، ویا جزء معروفی از یک **برنامه**. (مثلا يك تابع در ++C)

در واقع SCI ها برای **تشکیل اشیاء پیکربندی** سازماندهی می شوند که می توانند در پایگاه داده های پروژه با یک نام جداگانه به **فهرست** در آیند، شی پیکربندی دارای **نام و صفت** می باشد و از طریق یک سری **روابط** با سایر شی ها ارتباط پیدا می کند.



•SCI حاصل از محصولات کاری - ابزارها(مثل IDE و محیط برنامه نویسی و کامپایلر) جزئی از پیکربندی نرم افزارند.

•فلش خمیده رابطه ترکیبی را نمایش می دهد یعنی مدل داده و مولفه N از مشخصه های طراحی اند

- Software Configuration Items
- configuration objects

خطوط مبنا

خط مبنا عبارت است از یک مفهوم مدیریت وضعیت (پیکربندی) نرم افزار که به ما کمک می کند تا بدون این که توجه جدی به تغییر داشته باشیم، **تغییر را کنترل کنیم**.

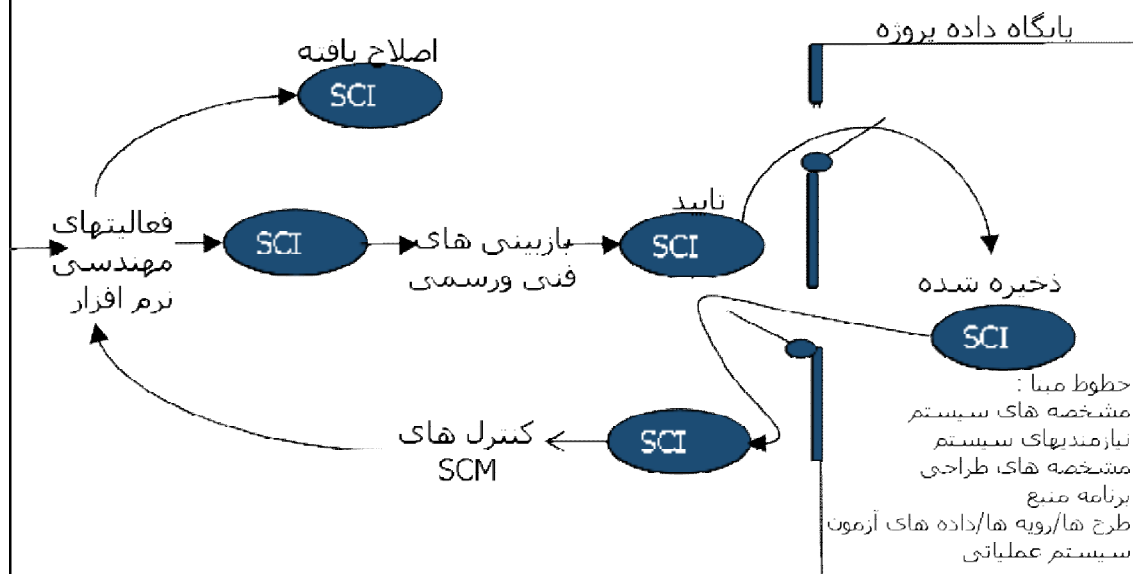
IEEE خط مبنا را به شرح زیر تعریف می کند:

یک ویژگی ویا محصولی که به طور اساسی مورد **بررسی قرار گرفته و به تأیید رسیده**، و پس از آن به عنوان پایه و اساسی برای **مراحل تکامل بعدی** مورد استفاده قرار خواهد گرفت و فقط می تواند به واسطه شیوه های اساسی کنترل تغییر، تغییر یابد. توصیف مبنا با مثال:

در یک رستوران از آشپزخانه به محیط رستوران ۲ در ، یکی برای ورود و یکی برای خروج وجود دارد اگر گارسون یک غذا را بردارد و قبل از اینکه از در خروج، خارج شود متوجه اشتباه بودن غذا شود بدون تشریفات و خیلی سریع و راحت آنرا تغییر می دهد. ولی اگر وی از در خارج شود، غذا را تحویل مشتری دهد و متوجه اشتباه خود شود بایست ۱- برای اطمینان از اشتباه به فیش نگاه کند ۲- از مشتری عذرخواهی کند ۳- از طریق در ورودی وارد آشپزخانه شود ۴- اشتباه را توضیح دهد و....

خط مبنا مثل درهای آشپزخانه هستند ، قبل از اینکه یک SCI (آیتمهای پیکربندی نرم افزار) به مبنا تبدیل شود تغییرات خاصی وجود ندارد و تغییرات غیر رسمی هستند ولی با عبور از خط مبنا ، تغییرات اساسی ایجاد می گردد

پیشرفت وقایع سبب ایجاد خط مبنا می شود، کارهای مربوط به مهندسی نرم افزار سبب ایجاد یک یا چند SCI می شوند. با بررسی و تأیید SCI ها، در یک پایگاه داده پروژه قرار می گیرند. وقتی که یکی از اعضای تیم مهندسی نرم افزار می خواهد بر روی SCI دارای مبنا تغییر ایجاد کند این SCI از روی پایگاه داده های پروژه بر روی فضای شخصی مهندسی کپی می شود. اما این SCI استخراج شده را تنها زمانی می توان تغییر داد که کنترل SCI ادامه پیدا کند.

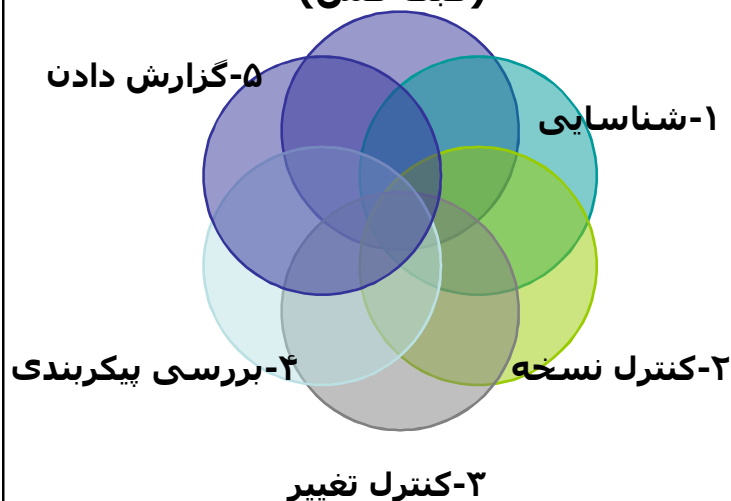


فرایند مدیریت پیکربندی نرم افزار (SCM)

مدیریت پیکربندی نرم افزار رکن مهمی از تضمین کیفیت نرم افزار است. مسئولیت اولیه آن، کنترل تغییر می باشد. اما SCM همچنین مسئول شناسایی SCI های جداگانه و نسخه های گوناگون نرم افزار، رسیدگی به پیکربندی نرم افزار برای حصول اطمینان از این که به طور مناسبی تکمیل شده و گزارش تمام تغییرات اعمال شده به پیکربندی. SCM به ۵ سوال زیر پاسخ می دهد:

- ۱- چگونگی سازگار شدن تغییرات در برنامه ها
- ۲- چگونگی کنترل نسخه نرم افزاری قبل از تحویل به مشتری
- ۳- چه کسی مسئول درجه بندی تغییرات است
- ۴- چگونگی اطمینان از انجام تغییرات
- ۵- مکانیزم ارزیابی تغییرات؟

پنج کار : SCM (کبک گش)



- (۱) شناسایی اقلام پیکربندی و ایجاد خط مبنا
- (۲) کنترل نسخه
- (۳) کنترل تغییر
- (۳-۱) کنترل دسترسی
- (۳-۲) کنترل همزمانی
- (۴) بررسی پیکربندی
- (۵) گزارش وضعیت

پنج کار SCM

شناسایی اشیاء در پیکربندی نرم افزار

برای کنترل و اداره قلم‌های پیکربندی نرم افزار، هر یک از آنها باید به طور جداگانه نام گذاری شده و سپس با استفاده از یک رهیافت "شی‌گرا" سازماندهی گردند. دو نوع شی قابل شناسایی هستند:

- ۱- **شی پایه:** عبارت است از **واحد متن** که توسط یک مهندس نرم افزار وبه هنگام تحلیل، طراحی، کددهی ویا آزمون ایجاد می شود.
- ۲- **شی مجتمع:** مجموعه ای از شی های پایه و سایر شی های مجتمع. ("ویژگی طراحی" یک شی مجتمع است)

هر شی دارای مجموعه ای از ویژگی های مشخص است که آن را به طور منحصر به فردی شناسایی می نماید: یک نام، یک توصیف، فهرستی از منابع، ویک تحقیق.

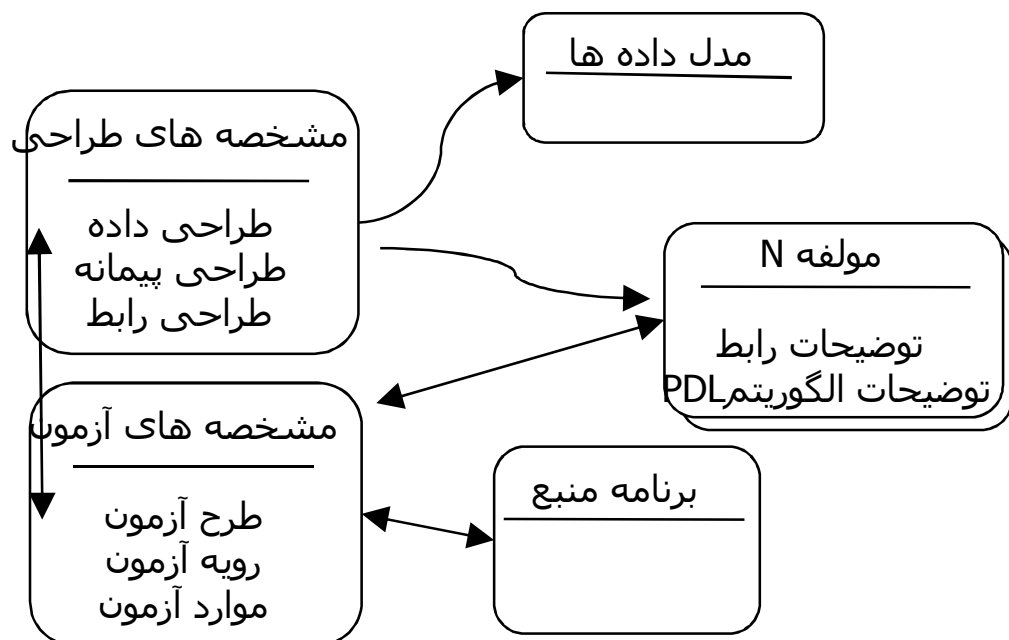
۱- **نام شی** عبارت است از یک رشته کاراکتر که شی را به طور مبهمی شناسایی می نماید.

۲- **توصیف شی** عبارت است از فهرستی از قلم های داده ها که موارد زیر را شناسایی می نماید:

- نوع SCI که بوسیله شی مشخص می شود
- شناسه پروژه

۳- **منابع** اطلاعات مربوط به نسخه ویا تغییر موجودیت هایی هستند که ارائه شده، پردازش شده، ارجاع داده شده ویا توسط شی درخواست می گردند.

۴- **تحقیق** نشانگری است در مورد "واحد متن" یک شی پایه وبی اعتبار در مورد یک شی مجتمع.



• شناسایی پیکربندی شی

شناسایی روابط بین اشیاست، دو نو رابطه بین اشیاء وجود دارد:

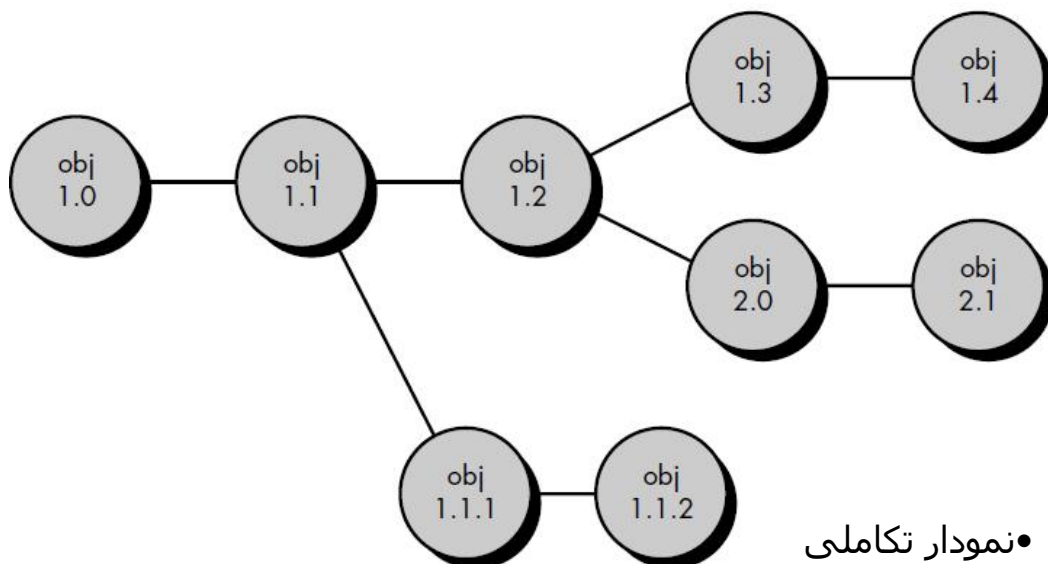
۱-ارتباط بین اشیاء مجتمع

۲-ارتباط بین یک شیء تجمعی (DataModel) و شیء پایه (Class m)

• روابط بین اشیاء

روابط بین شیء های پیکربندی را می توان با استفاده از یک زبان برهم بندی پیمانه ای (MIL) نشان داد. یک وابستگی بین شیء های پیکربندی را توصیف می کند و کمک می کند تا هر نوع نسخه ای از یک سیستم به طور خودکار ساخته شود.

شیء ها در طول فرایند نرم افزاری به وجود می آیند. پیش از آنکه یک شیء به صورت خط مبنا درآید و حتی پس از این که یک خط مبنا ایجاد گردد ممکن است چندین بار تغییر کند. می توان برای هر شکل یک نمودار تکاملی ایجاد کرد. نمودار تکاملی تاریخچه تغییر شیء را توصیف می کند.



• بازبینی ها باعث تبدیل شیء 1.0 به شیء 1.1 می شوند و انجام اصلاحات جزئی و تولید نسخه ۱.۱.۱ و ۱.۱.۲ و با اصلاحات عمده ، نسخه ۱.۲ ایجاد می شود و

• نمودار تکاملی module interconnection language

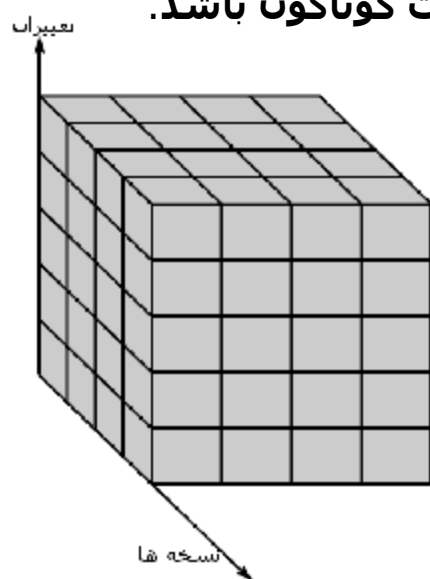
کنترل نسخه

کنترل نسخه ؛ روش ها و ابزارها را ترکیب می نماید تا نسخه های گوناگونی از شی های پیکربندی را که طی فرایند نرم افزار ایجاد شده اند مهار نمایند.

"آقای کلم" کنترل نسخه در متن SCM را به شرح زیر توصیف می کند:

مدیریت پیکربندی به یک کاراین امکان را می دهد تا گزینه های پیکربندی گوناگونی از سیستم نرم افزاری را، از طریق انتخاب نسخه های مناسب مشخص نماید. این امر از طریق ربط دادن صفات با هر یک از نسخه های نرم افزاری و سپس مشخص نمودن یک پیکربندی از طریق توصیف مجموعه ای از صفات دلخواه، پشتیبانی می گردد.

- یکی از راه های نمایش نسخه های گوناگون یک سیستم نمودار تکاملی است.
- هر یک از گره های موجود در نمودار، یک **شی مجتمع**، یعنی **نسخه کاملی از نرم افزار**، می باشند.
- هر نسخه نرم افزار عبارت است از **مجموعه ای از SCI ها و هر نسخه می تواند ترکیبی از تغییرات گوناگون باشد**.



- روش دیگر مفهومی نمودن رابطه بین موجودیت ها، متغیرها و نسخه ها عبارت است از ارائه آنها به عنوان یک **مخزن اشیاء**.

- در **مخزن اشیاء** رابطه بین موجودیت ها، متغیرها و نسخه ها می تواند به صورت فضایی سه بعدی نمایش داده شود و یک موجودیت تشکیل شده است از مجموعه ای از شی ها که نسخه تجدیدنظر شده آنها در یک سطح قرار دارند. (عناصر موجودیت ورودی)
- یک متغیر (حالت تغییر) **مجموعه متفاوتی از شی ها است** که نسخه تجدیدنظر شده آنها در یک **مجموعه وجودی** ها سطح است و بنابراین در موازات سایر متغیرها قرار دارد.

چندین روش خودکار گوناگون برای کنترل نسخه پیشنهاد و ارائه شده؛ تفاوت اصلی این شیوه ها در پیچیدگی صفاتی است که برای ساخت نسخه ها و متغیرهای خاصی از یک سیستم و روش کار فرایند ساخت به کار می رود

شی های پایه و شی های مجتمع، تشکیل یک مخزن اشیاء را می دهند که نسخه ها و گونه های متفاوت دیگر از روی آن ساخته می شوند. کنترل نسخه عبارت است از مجموعه ای از شیوه ها و ابزارها که برای کنترل استفاده از این شی ها به کار می روند

کنترل تغییرات

به عقیده جیمز باج کنترل تغییر خیلی زیاد باعث ایجاد مشکلات می شود و کنترل تغییر خیلی کم هم مشکلات دیگری را ایجاد می کند.

یک درخواست تغییر، برای ارزیابی قابلیت فنی، تأثیرات جانبی بالقوه، تأثیر کلی بر روی سایر شی های پیکربندی و عملکرد سیستم، و هزینه پیش بینی شده تغییر، ارائه و ارزیابی می گردد. نتایج ارزیابی به صورت یک گزارش تغییر ارائه می گردد که توسط یک **مجوز کنترل تغییر (CCA)** مورد استفاده قرار می گیرد. CCA فرد یا گروهی است که تصمیم نهایی را در مورد وضعیت و اولویت تغییرات می گیرد برای هر تغییر به تأیید رسیده، یک **دستور (ترتیب؟؟؟) تغییر مهندسی (ECO)** ایجاد می شود. ECO تغییری را که قرار است به وجود آید توصیف می نماید؛ محدودیت هایی که باید اعمال شوند، و معیارهایی برای مطالعه و بررسی، شی که باید تغییر کند از پایگاه داده های پروژه بیرون کشیده شده، تغییر اعمال شده و فعالیت های SQA مناسبی اعمال می گردند. سپس شی وارد پایگاه داد ها می شود.

□ فرایند "بیرون کشیدن" و "وارد نمودن" (از بانک اطلاعاتی) دو رکن اصلی کنترل تغییر را ایجاد می کند:

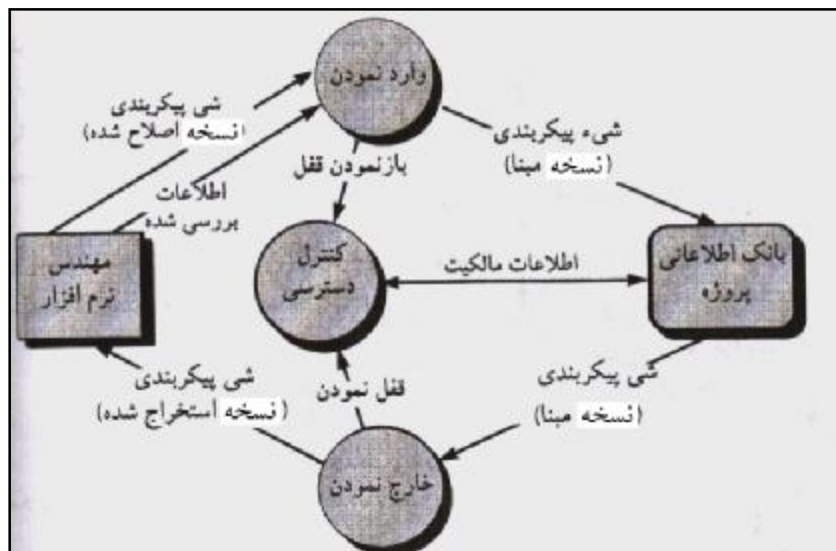
۱. کنترل دستیابی :

تعیین می کند کدام مهندس نرم افزار اختیار ارزیابی و اصلاح یک شی پیکربندی خاص را دارند. (زمانی که یک قلم پیکربندی به عنوان خط مبنا در مخزن ذخیره می شود و فردی درخواست تغییر می دهد مدیر باید بررسی کند که آیا او اجازه دسترسی به این قلم را دارد یا خیر)

۲. کنترل هم زمانی:

کمک می کند اطمینان حاصل شود که تغییرات مشابهی که توسط دو فرد مختلف انجام می شوند؛ بر روی هم کپی نشوند. (دو نفر به طور همزمان نمی توانند یک قلم پیکربندی را تغییر دهند. برای این منظور با هر دسترسی به یک قلم، آن قلم باید قفل شود.)





• کنترل دسترسی و همزمانی

- یک مهندس نرم افزار بر اساس درخواست تغییر تأیید شده و ECO یک شیء پیکربندی را بیرون می کشد.
- عمل ارزیابی کنترل به ما اطمینان می دهد که مهندس نرم افزار اختیار بیرون کشیدن شیء را دارد و هم زمانی کنترل شیء را در پایگاه داده های پروژه نگه می دارد، به طوری که این شیء به هیچ عنوان به روز نمی شود تا زمانی که نسخه ای که به تازگی بیرون کشیده شده جایگزین آن گردد.
- یک نسخه از شیء دارای مینا، که نسخه استخراج شده نامیده می شود، توسط مهندس نرم افزار اصلاح می گردد.
- پیش از آنکه یک SCI تبدیل به یک خط مینا گردد، تنها نیاز به استفاده از کنترل تغییر غیر رسمی وجود دارد.
- برای ایجاد تغییر، چنانچه تغییر سایر SCIها را تحت تأثیر قرار دهد، سازنده باید از مدیر پروژه ویا از CCA کسب اجازه نماید.

• فرایند کنترل تغییر:

- ۱- نیاز به تغییر تشخیص داده می شود ۲- درخواست تغییر از کاربر می رسد ۳- سازنده ارزیابی می کند ۴- گزارش تغییر تولید می شود ۵- فرد مجاز به کنترل تغییر تصمیم می گیرد : A. درخواست تغییر پذیرفته نمی شود و در مرحله بعد کاربر مطلع می شود. B. درخواست تغییر در نوبت اقدام قرار می گیرد و ECO تولید می شود:
- افراد به اشیاء پیکربندی نسبت داده می شوند- اشیاء (اقلام) پیکربندی خارج می شوند- نیاز به تغییر تشخیص داده می شود- تغییر اعمال می شود- تغییر مورد واریسی قرار می گیرد- اقلام تغییر یافته پیکربندی، وارد می شوند- خط مبنایی برای آزمون برقرار می شود- فعالیت های تضمین کیفیت و آزمون انجام می شود- تغییرات برای ارتقاء نسخه بعدی مستقر می شود- تغییر در نسخه جدید قرار می گیرد- نسخه جدید توزیع می شود
- مسئول کنترل تغییر (CCA) نقش فعالی در لایه های دوم و سوم کنترل، ایفا می کند. نقش CCA عبارت است از داشتن یک دیدگاه جهانی؛ سوالاتی مانند اینکه تغییر چگونه می تواند درک مشتریان را نسبت به محصول اصلاح نماید؟ توسط CCA مورد بررسی قرار می گیرند.

وارسی پیکربندی

- شناسایی، کنترل نسخه و کنترل تغییر به سازنده نرم افزار کمک می کند تا ترتیب را برقرار نماید. و از آشفتگی جلوگیری کند مکانیزم های کنترل فقط تا زمانی تغییر را دنبال می نمایند که یک ECO ایجاد گردد.
- برای کسب اطمینان از پیاده سازی مناسب تغییر؛ از طریق ۱- بررسی های فنی رسمی ۲- وارسی پیکربندی نرم افزار عمل می کنیم.
- **بررسی فنی رسمی: به صحت فنی شی پیکربندی که اصلاح** گردیده می پردازد. SCI ها و نارسایی ها و یا اثرات جانبی بالقوه را ارزیابی می نماید. بررسی فنی رسمی باید در مورد تمام وحتی جزئی ترین تغییرات صورت پذیرد.
- **وارسی پیکربندی نرم افزار:** بررسی فنی رسمی را از طریق ارزیابی ویژگی های یک شی پیکربندی که معمولاً طی بررسی در نظر گرفته نمی شود. تکمیل می نماید
- در وارسی یک سری سوالات برای کسب اطمینان پرسیده می شود
- **نتیجه:** وارسی پیکربندی عبارت است از یک فعالیت SQA که به حصول اطمینان از کیفیت به هنگام اعمال تغییرات کمک می نماید

گزارش وضعیت

- گزارش وضعیت پیکربندی که به آن **صورت وضعیت (حسابرسی وضعیت)** هم می گویند؛ یکی از وظایف SCM است.
- جریان اطلاعات در مورد گزارش وضعیت پیکربندی را به اختصار CSR می نامند. هر بار که یک SCI هویت جدید و یا به روزی را انتخاب می نماید، یک مورد CSR ایجاد می شود. هر بار که وارسی پیکربندی صورت می پذیرد، نتایج (وارسی پیکربندی) به عنوان بخشی از کار CSR گزارش میگردند. خروجی CSR در یک پایگاه داده ممکن است روی خط قرار بگیرد، که در این صورت سازندگان یا اداره کنندگان نرم افزار می توانند اطلاعات مربوط به تغییر را به وسیله کلید واژه ارزیابی نمایند.
- یک گزارش CSR بر طبق یک مبنای ثابت ارائه می شود و هدف آن این است که مدیر و سازندگان به ارزشیابی تغییرات ادامه دهند.
- گزارش وضعیت پیکربندی نقش مهمی در موفقیت پروژه های بزرگ نرم افزاری ایفا می کند.
- **نتیجه:** گزارش وضعیت، اطلاعاتی را درباره هر تغییر به کسانی که نیازمند آگاهی از آن هستند ارائه می نماید



استانداردهای مدیریت پیکربندی نرم افزار

- بسیاری استانداردهای مدیریت پیکربندی نرم افزار بیش از دو دهه است که ارائه می شود. بسیاری از استانداردهای اولیه بر توسعه و ساخت نرم افزارهای نظامی متمرکز شده و تاکید داشته اند. استانداردهای ANSI/IEEE برای نرم افزارهای غیرنظامی به کار می رود و برای سازمانهای مهندسی نرم افزار کوچک و بزرگ توصیه می شود.
- استانداردهای اولیه مانند:
MIL-STD-483 ، DOD_STD_480A ، و MIL-STD-1521A
- استانداردهای ANSI-IEEE مانند:
ANSI/IEEE Stds. NO.828-1983 ، NO.1042-1987

فصل دهم : مهندسی سیستم

□ مهندسی نرم افزار در نتیجه فرایندی بنام مهندسی سیستم ایجاد می گردد.

□ مهندسی سیستم (دیدگاه پرسمن) :

۱. مهندسی فرایند تجاری (Business Process Engineering) :

چنانچه محدوده کار مهندسی تمرکز بر روی کار و تجارت داشته باشد ، مهندسی سیستم را مهندسی فرایند کاری می نامند.

۲. مهندسی محصول (Product Engineering) :

زمانی که محصولی ایجاد می گردد، فرایند مهندسی سیستم را مهندسی محصول می نامیم .

□ مهندسی سیستم در مورد چه چیزی بحث می کند؟

پیش از آنکه نرم افزاری طرح ریزی و تهیه گردد باید سیستمی که نرم افزار در آن قرار داده می شود مورد مطالعه قرار گیرد. باید هدف کلی سیستم مشخص گردد و نقش سخت افزار، نرم افزار، پایگاه داده ها، پردازش ها و دیگر اجزاء سیستم شناخته شده و نیازهای عملکردی آن بررسی، تحلیل و مشخص گردد.

□ کار مهندس سیستم چیست؟

سعی مهندسان سیستم بر آن است تا از راه تماس با خریداران، کاربران آینده و دیگر افراد ذریط سیستمی ایجاد شود که پاسخگوی نیاز آنها باشد.

□ اهمیت مهندسی سیستم چیست؟

يك ضرب المثل مي گوید «جنگل را نمي توان قبل از دیدن درختان دید»

• جنگل = خود سیستم

• درختان = ارکان فناوری

ولي چنانچه پیش از فهم سیستم به ساخت اجزاء فنی پرداخته شود بی شک دچار اشتباهاتی خواهیم شد. پس

پیش از اندیشیدن درباره درختان ، باستی به درک جنگل پرداخت.

□ مراحل شناخت سیستم (درک جنگل) کدام اند؟

اهداف و جزئیات بیشتر درباره نیازمندیهای عملیاتی را می توان با ۱- آگاهی از خواسته ها و نیازهای خریداران کسب نمود. سپس

۲- نیازمندیها تحلیل شوند تا از میزان وضوح، جامع بودن و قابلیت انطباق و ثبات آنها تعیین گردد.

۳- سرانجام تمامی کارهای لازم جهت برآورده شدن نیازمندیهای مذکور صورت می گیرد تا اطمینان حاصل شود که تغییرات به بهترین نحو انجام شده است.

□ حاصل کار چیست؟

تهیه الگویی کارآ از سیستم بایستی منجر به پدید آمدن مهندسی سیستم گردد. کار نمونه سازی می تواند تعیین خصوصیات سیستم مورد نظر و یا حتی تهیه نمونه ای نمادین از آن باشد

□ چگونه از درستی انجام کارها مطمئن گردیم؟

کار تولید شده توسط سیستم باید از نظر میزان وضوح، جامع بودن، قابلیت انطباق و ثبات مورد بررسی مجدد قرار گیرد. چنانچه انجام تغییراتی دیگر ضروری باشد آن تغییرات بایستی با استفاده از SCM سه بعدی صورت گیرد.

سیستمهای مبتنی بر کامپیوتر

• **تعریف سیستم** در واژه نامه Websters:

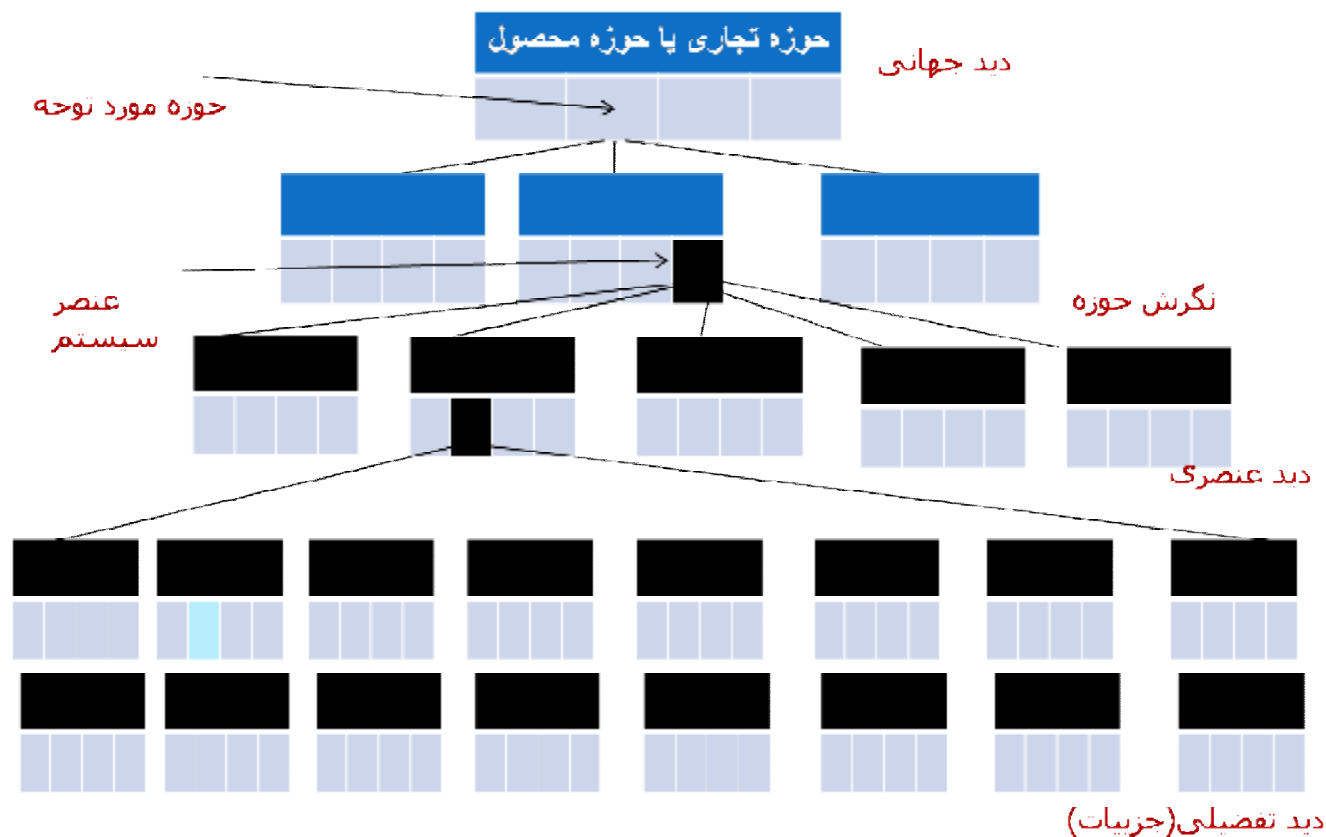
۱. مجموعه و یا اجزاء آرایشی که چنان پیوندی با هم دارند که شکلی واحد و یا کل سازمانی را تشکیل دهند.
۲. چند حقیقت، اصول، قاعده و غیره که چنان طبقه بندی شده و نظم یافته اند تا شکلی سازمان یافته با پیوندهای منطقی موجود میان اجزاء را نشان دهند.
۳. روش یا نقشه ای از نظم و ترتیب و یا رده بندی موجود.
۴. راهی برای انجام چیزی، روشی، رویه و....

□ **تعریف سیستم کامپیوتری**

مجموعه یا آرایشی از اجزاء که به گونه ای سازمان یافته اند تا برخی اهداف از پیش تعریف شده را از راه پردازش اطلاعات انجام دهد. این هدف ممکن است شامل برخی امور تجاری یا ارائه محصولی برای فروش باشد

□ **ارکان مختلف سیستم:**

۱. **نرم افزار:** شامل برنامه های کامپیوتر، ساختار داده ها و ارائه مستندات مربوطه می باشد که در انجام روش، روند و یا کنترل منطقی اعمال مورد نظر موثر است.
۲. **سخت افزار:** ابزاری الکترونیکی است که قابلیت انجام محاسبات را ممکن ساخته است. (سویچ/سنسور/موتور و ...)
۳. **افراد:** عبارتند از کاربران و استفاده کنندگان سخت افزار و نرم افزار.
۴. **پایگاه داده:** مجموعه وسیع و سازمان یافته اطلاعات که از راه نرم افزار قابل دسترسی است.
۵. **مستندات:** اطلاعاتی توصیفی است که راه استفاده و یا نحوه کارکرد سیستم را می نمایند.
۶. **رویه ها:** مراحل است که کاربرد ویژه هر یک از ارکان سیستم و یا متن رویه ای موجود در سیستم مورد نظر را توضیح می دهد.



سلسله مراتب مهندسی سیستم

- سلسله مراتب در مهندسی سیستم از بالا به پایین و نیز از پایین به بالاست.
- روند مهندسی سیستمها با چشم انداز جهانی آغاز می گردد. سپس دایره چشم انداز جهانی را آنقدر تنگ می کنند تا هر چه بیشتر بر یک جنبه ویژه از خصوصیت مورد نظر، کانون توجه قرار گیرد. سرانجام مرحله های تحلیل، طراحی و ساخت سیستم مورد نظر انجام می شود.
- در بالاترین رده سلسله مراتب، موضوعهای بسیار کلی و در پایین ترین رده، جزئیات کارهای تکنیکی قرار دارد.

$$WV = \{D1, D2, \dots, Dn\}$$

world view (WV) = Set of domains (D_n)

$$D_i = \{E1, E2, \dots, E_m\}$$

Domain (D_i) = Set of elements (E_m)

$$E_j = \{C1, C2, \dots, C_k\}$$

elements (E_j) = Set of technical components (C_k)

مدلسازی سیستم

مهندسی سیستم عبارت است از فرآیند مدل سازی.

خصوصیات مدل‌های تهیه شده بدون تمرکز بر روی دیدگاه کلی یا جزئی:

- فرآیند مناسب جهت برآورده ساختن نیازهای دیدگاه مورد نظر تعیین گردد.
- چگونگی رفتاری فرآیندها و فرض‌هایی را که رفتارها بر آن اساس پایه گذاری شده اند بیان کند
- به روشنی ورودی درون رو و برون رو مدل (ارتباط با هم سطحی‌ها یا با سطوح دیگر) مربوطه تعریف شود.
- تمامی پیوندها (ازجمله خروجی) که درک بهتر دیدگاه مورد نظر را برای شخص مهندس میسر می سازد تشریح و نمایانده شود.

عوامل مهار کننده در ساختن مدل سیستم

در ساختن مدل سیستم مهندس باید چندین عامل مهار کننده (بازدارنده) را در نظر گیرد:

۱. **فرضیات (مفروضات):** شمار تبدیلات و متغیرهای ممکن را کاهش می دهد و در نتیجه مدل تهیه شده می تواند مشکلات موجود را به شیوه ای منطقی بازتاب دهد. (مثلاً: عدم پیچیده شدن پا دور بدن انسان باعث کاهش حالات در بازیهای رایانه ای می شود)
۲. **ساده سازی:** امکان می دهد مدل طبق برنامه ریزی زمانی و به موقع آماده گردد. (مثلاً : به حداقل رساندن تعداد منابع ورودی)
۳. **محدود سازی:** محدوده سیستم مورد نظر را تعیین می کند. (مثلاً: با تعیین هدف ، حوزه های دیگر نیز محدود می شوند...)
۴. **تنگناها:** سبب می گردد تا هنگام ساخت مدلها رهیافت و تصمیمات مناسبی اتخاذ گردد. (مرز امکانات)
۵. **اولویتها:** ساختار مطلوب فناوری و عملیات داده ها را تعیین می نماید.

مدل سیستم بدست آمده با هر دیدگاه می تواند به يك راه حل خودکار کامل، راه حل نیمه خودکار و یا روش غیرخودکار منتهی شود. در نتیجه مهندسی سیستم ، فقط ارتباط نسبی بین عناصر مختلف سیستم را اصلاح می نماید تا مدلهایی از هر نوع را بدست آورد



شبیه سازی سیستم

- سالها پیش روش ساخت سیستمها مانند هواپیما ساختن برادران رایت بود. وسیله ای را بطور کامل می ساختیم و بعد آن را بکار می بردیم، خراب می شد و دوباره از ابتدا ساخته می شد.
- امروزه ابزارهای نرم افزاری جهت شبیه سازی و مدل سازی سیستم در راه رفع موارد غافلگیرکننده در ساخت سیستمهای واکنشی مبتنی بر کامپیوتر به خدمت گرفته شده اند.
- مهندس سیستم توسط ابزارهای مدل سازی و شبیه سازی می تواند خصوصیات یک سیستم را مورد آزمون قرار دهد.

مهندسی فرآیند تجاری (BPE) business process engineering

- هدف مهندسی فرآیند تجاری توضیح معماریهای است که یک حرفه را قادر می سازد تا از اطلاعات استفاده بهینه کند.
- هنگامی که نیازهای فناوری اطلاعات شرکتی را از دیدگاه جهانی مورد بررسی قرار می دهیم دیگر در وجود نیاز به مهندسی سیستم هیچ شکي باقی نمی ماند. در این راه هم آگاهی از خصوصیات و جزئیات ساختار مناسب محاسباتی مورد نیاز است و هم آنکه ساختار نرم افزار که اشکال گوناگون و منحصر به فرد منابع نامتقارن محاسباتی را در خود جای می دهد، باید ارتقاء یابد.
- مهندسی فرآیند تجاری روشی برای ایجاد طرحی فراگیر جهت پیاده سازی معماری محاسباتی می باشد.

□ جهت حصول دیدگاه کلی از نیازهای تکنولوژی یک شرکت، از مهندسی سیستم استفاده می نمایم. برای این منظور سه معماری مختلف می بایست در حیطه اهداف تجاری تحلیل و طراحی گردد:

۱. معماری داده ها
۲. معماری کاربردها
۳. فناوری فراساختاری

□ معماری داده ها

چارچوبی را جهت اطلاعات مورد نیاز یک رشته تجاری و یا جهت انجام عملیات تجاری فراهم می سازد. بلوکهای ساختاری جدا جدا این معماری، داده هایی هستند که در زمینه تجاری کاربرد دارند. یک شیء داده ای دربردارنده تعریف یک رشته صفات خاصه آن، از برخی جنبه ها، کیفیت، خصوصیت، یا دیگر ویژگی هایی است که وصفی از داده مورد بحث ارائه می دهد.

هر بار که مجموعه اشیاء داده تعریف می شود، رابطه ها پیوندهای میان آنان نیز بررسی و بیان می گردد. «پیوند موجود» چگونگی ارتباط یک شیء را با دیگر اشیاء شرح می دهد. یعنی یا خریدار کالای A را می خرد و یا کالای A توسط خریدار، خریداری می شود.

□ معماری کاربرد

شامل **عناصر یک سیستم** است که **اشیاء** را به منظور رسیدن به برخی **اهداف** تجاری به روند معماری داده ها انتقال می دهد. معماری کاربرد نقش افراد و روندهای تجاری را که هنوز خودکار نشده اند شامل می شود.

□ فناوری فراساختاری

«زیرساخت فناوری» پایه ریزی داده ها و معماری کاربرد را میسر می سازد. **زیرساختها دربرگیرنده سخت افزار و نرم افزارهایی است که جهت پشتیبانی کاربردها و داده ها به کار گرفته می شود.** و شامل کامپیوترها ؛ سیستم عاملها ؛ اتصالات و است

□ مدل سازی معماری های سیستم

برای مدل سازی معماری های سیستم ، سلسله مراتبی از فعالیت های مهندسی فرآیند تجاری تعریف می شود. با توجه به شکل مقابل میتوان دریافت که **دیدگاه جهانی** برگرفته از «**برنامه ریزی راهبرد اطلاعات**» **ISP** است. این برنامه ریزی تمامی امور تجاری را به عنوان ورودی در نظر گرفته و حوزه های مختلف آن را از یکدیگر مجزا می کند. حوزه این دیدگاه موسوم به فعالیت تحلیل ناحیه ای تجاری «**BAA**» است.

BAA جزئیات انواع داده ها و نیز اثرهای متقابل را

تعین می نماید. **BAA** تنها به امر ارائه تعریف هایی که

در زمینه تجاری مورد نیاز است سروکار دارد. در این

دیدگاه مهندس سیستم کار خود را به حوزه مشخصی

محدود می کند در دیدگاه **BAA** حوزه تجاری یک واحد کل

در نظر گرفته می شود و عملکردها و روندهای آن را که

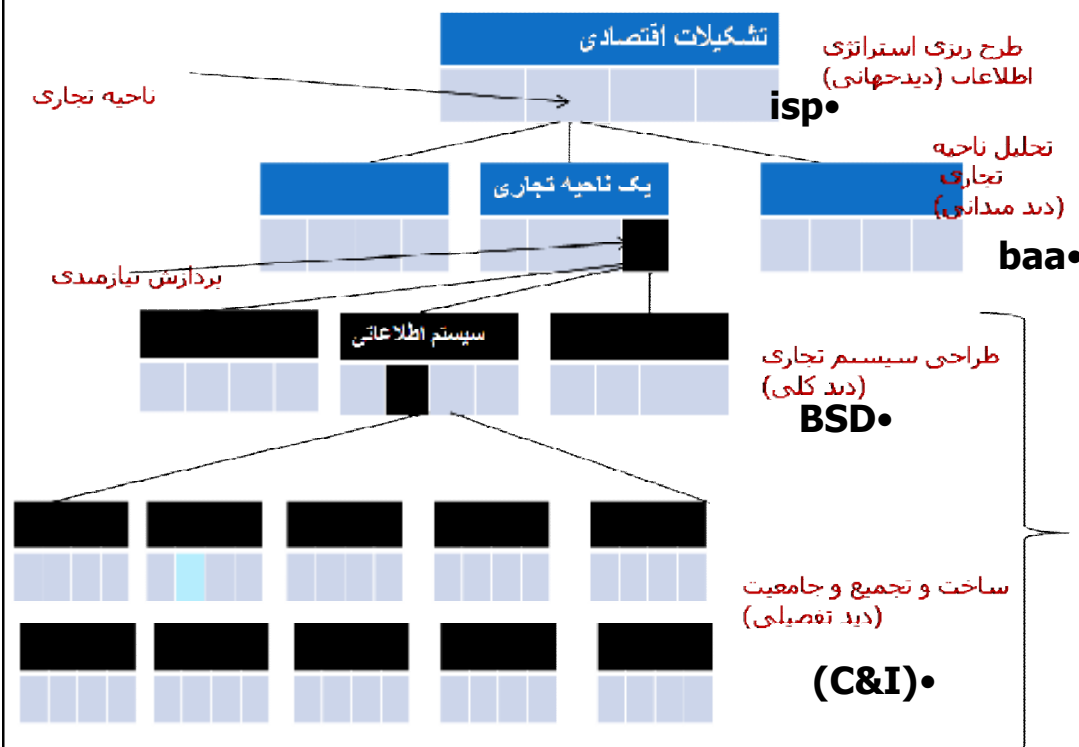
امکان می دهد تا تاجر بتواند به واسطه آنها نیازهای

خود را برآورده سازد و به اهدافش دست یابد، مجزا می

کند. برآیند **BAA** عبارت است از امکان جداسازی حوزه ای

که در آن سیستمهای اطلاعات می توانند از حوزه

تجاری پشتیبانی کنند.



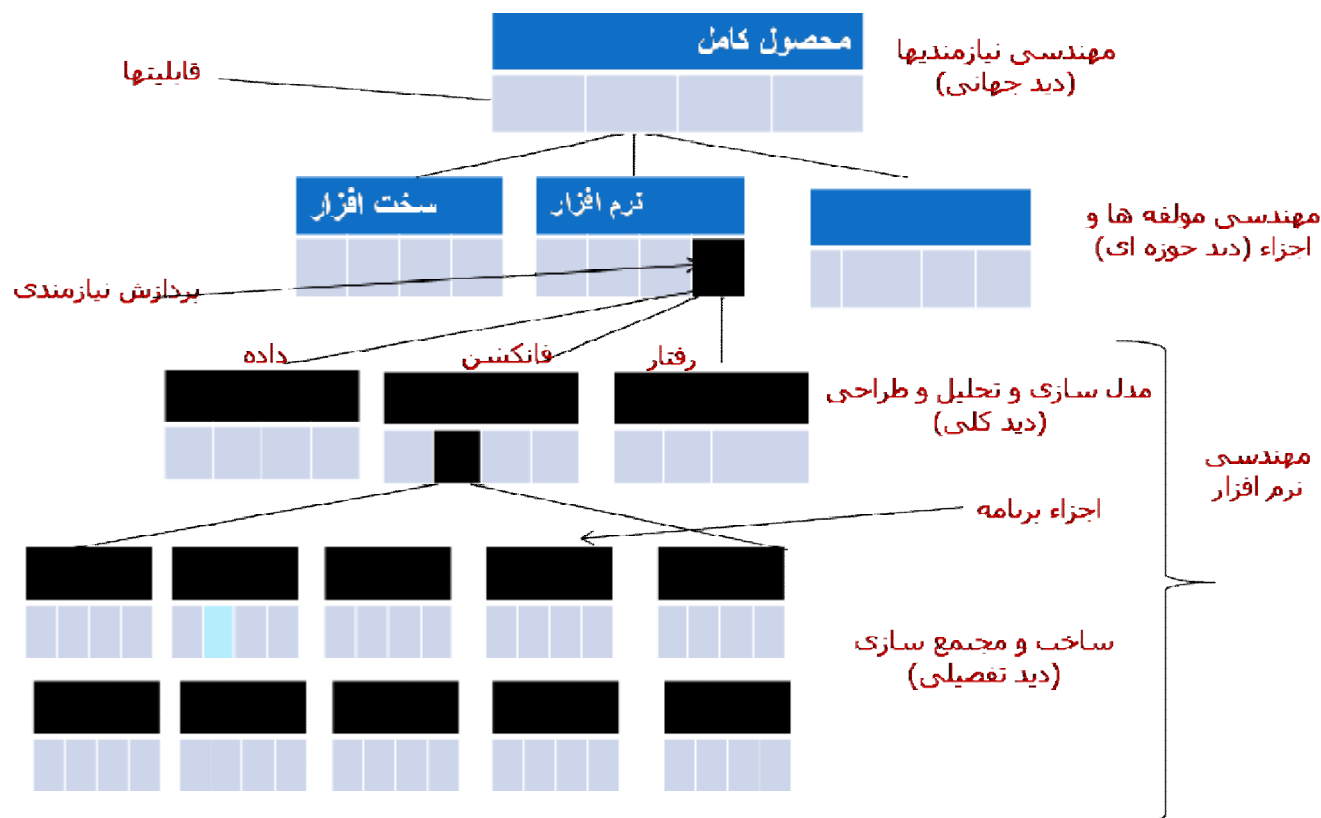
information strategy planning (isp)

business area analysis (baa)

- وقتی یک سیستم اطلاعات جهت پیشرفت بیشتر ایزوله می شود، مسیر BPE وارد **مرحله مهندسی نرم افزار می گردد**. با فراخواندن مرحله « طراحی سیستم تجاری » (BSD)، **نیازهای اساسی سیستم اطلاعات مورد نظر مشخص و طراحی می شوند**، سپس نیازمندیهای مذکور به موضوعات معماری داده ها، معماری کاربرد و زیرساختهای فناوری تبدیل می شوند.
- در مرحله نهایی BPE یعنی مرحله «**ساختمان و مجتمع سازی**» (C&I) به **تکمیل و انجام جزئیات پرداخته می شود**.

مهندسی محصول

هدف مهندسی محصول این است که **خواستههای خریداران در مورد یک رشته از قابلیتهای تعریف شد را به محصول بالفعل تبدیل نماید**. معماری چهار جزء مشخص و مجزای سیستم را در بر می گیرد که عبارتند از:



- 1- نرم افزار
- 2- سخت افزار
- 3- داده ها
- 4- افراد

- در شکل مقابل **دیدگاه کلی** از طریق **مهندسی نیازمندیها** حاصل می شود. تمام اطلاعات مربوط به خواسته ها و نیازمندیهای را که خریداران انتظار دارند از طریق خود آنها کسب می گردد.
- با شروع مهندسی انسانی و مهندسی پایگاه داده اجزا سیستم، مرحله **تعیین عملکردها و رفتارها** نیز صورت می گیرد
- **عنصر دید کلی** (دیدگاه) برای مهندسی محصول، روش مهندسی است که برای تحلیل و طراحی بکار گرفته می شود
- **فعالیت در زمینه ساخت و مجتمع سازی** نیز شامل تولید برنامه، آزمون و مراحل پشتیبانی است

مهندسی نیازمندیها

بهترین راه حل برای کسب اطمینان از قدرت سیستم در برآورده شدن نیازهای مشتری ، مهندسی نیازمندیها است
مهندسی نیازمندیها مکانیزم مناسبی را جهت درک خواسته های خریداران فراهم می سازد و نیز امکان تحلیل آن نیازها، برآورد امکانات، دستیابی به راه کارهای منطقی، بیان راه حل مذکور به طور روشن و واضح، ارزیابی خصوصیات و مدیریت نیازمندیها هنگامی که به سیستم عملیاتی تبدیل می شوند را فراهم می آورد.
فرآیند مهندسی نیازمندیها شامل شش قسمت زیر است:

- | | |
|---------------------------------|--|
| ۱-مرحله یافتن نیازمندیها | ۲-مرحله تحلیل نیازمندیها و انجام بحث و گفتگو درباره آنها |
| ۳-مرحله تعیین مشخصات نیازمندیها | ۴-مرحله مدل سازی سیستم |
| ۵-مرحله ارزیابی نیازمندیها | ۶-مرحله مدیریت نیازمندیها |

تعیین نیازمندیها

ظاهرا تعیین نیازمندیهای خریداران ساده به نظر می رسد اما حقیقت آن است که پاسخ به سوالات مورد نظر دشوار است. چند مشکل موجود عبارتند از:

مشکلات موجود در دامنه دید مسائل(محدوده): تعریف حدود سیستم ناقص صورت می گیرد و یا اینکه خریدار به ذکر جزئیات تکنیکی غیرضروری می پردازد. که می تواند باعث سردرگمی کل اهداف سیستم شود.

مشکلات موجود در زمینه درک مسائل(مفهومی): کاربرها یا مشتری ها از موارد مورد نیاز خود آگاهی چندانی ندارند. مطمئن نیستند. اطلاعاتی را حذف می کنند که به عقیده آنها واضح است و نیازهایی را مشخص می کنند که مبهم یا غیرقابل آزمایش هستند

دشواری ارزیابی: نیازها در طول زمان تغییر می کنند.

مهندسان سیستم جهت فایق آمدن بر مشکلات فوق باید فعالیتهای مربوط به جمع آوری اطلاعات در زمینه نیازهای خریداران را به شیوه ای سازمان یافته انجام دهند

❑ رهنمودهای سامرویل و ساویر برای یافتن نیازمندیها

- ❖ تخمین کار و امکان سنجی تکنیکی سیستم مورد نظر
- ❖ شناخت افرادی که به مشخص کردن نیازمندیها و نیز به درک خواسته های خریداران و تمایل به سازمان یافتگی، کمک می کنند.
- ❖ تعریف کردن محیط تکنیکی برای عرصه ای که سیستم یا محصول در آن جای می گیرد.
- ❖ تعریف کردن «حوزه محدودیت ها» که دایره عملیات و یا کارکرد سیستم یا محصولی که قصد ساختن آن را داریم، محدود می کند.
- ❖ تعریف و تعیین یک یا چند روش جهت یافتن نیازمندیها
- ❖ از افراد زیادی جهت شرکت در نشست ها دعوت به عمل آید تا نیازمندیها از دیدگاههای مختلف تعریف و بررسی گردد.
- ❖ نیازمندیهای دارای ابهام برای ساخت نمونه اولیه در نظر گرفته می شود.
- ❖ سناریو کاربرد پدید آورده می شود تا خریداران/کاربران بتوانند نیازمندیهای کلیدی خود را بشناسند.

❑ محصول کاری(نقاط عطف) در سیستم بدست آمده در نتیجه فعالیت دریافت نیازها:

- ❖ نیازمندیها و قابلیت برآورده ساختن آنها تصریح شود
- ❖ حیطه سیستم و یا محصول تعیین گردد
- ❖ فهرستی از مشتري ها، کاربران و... که در دریافت تیاها موثرند ایجاد شود
- ❖ محیط تکنینی سیستم تعریف و توصیف شود
- ❖ فهرستی از نیازمندیها ارائه شده و حوزه محدودیت هایی که در مورد هر یک از آنها اعمال می شود بیان گردید.
- ❖ تهیه یک رشته سناریوی استفاده که امکان مشاهده سیستم مورد استفاده یا محصول را زیر شرایط مختلف عملکرد فراهم می آورد
- ❖ هر یک از طرح های اولیه را بایستی ارتقاء داد تا نیازمندیهای مطرح شده تعریف بهتر و روشن تری داشته باشند.

تحلیل نیازمندیها و مذاکرات مربوطه

پس از جمع آوری اطلاعات، در مرحله «تحلیل نیازها» نیازمندیها **دسته بندی** شده و به صورت **مجموعه ها و زیرمجموعه های** مربوطه در می آیند، **رابطه نیازها** بایکدیگر بررسی می شود، آنها از نظر ثبات و سازگاری، قابلیت حذف شدن، و عدم وجود ابهام **سنجیده و مورد آزمون قرار می گیرد و سرانجام اولویت نیازمندیها بر اساس خواسته ها و انتظارات خریداران/کاربران تعین می گردد.**

• پرسشهایی که در حین فعالیت تحلیل نیازمندیها باید پاسخ داده شود:

- آیا هر نیازمندی با **هدف کلی سیستم** و یا **محصول مورد نظر سازگاری** دارد؟
 - آیا تمامی نیازمندیها بطور **مختصر و مفید** مطلب مورد نظران را بیان می دارند؟
 - آیا نیازمندیهای مطرح شده **برآستی ضرورت رفع آن را احاط می کند** و یا با تحقق آن صرفا خصوصیتی به سیستم اضافه می گردد که **جزء اهداف ساخت آن سیستم نیست**؟
 - آیا محدوده هر نیازمندی **مشخص و بدون ابهام** است؟
 - آیا هر نیازمندی **موارد استناد** نیز دارد، بدین معنی که آیا منبعی برای هر نیاز در نظر گرفته شده است؟
 - آیا نیازمندی ای هست که با دیگر نیازمندیها **مغایرت** داشته باشد؟
 - آیا می توان به هر یک از نیازمندیها در **محیط تکنیکی** که سیستم تولید شده در آن جای می گیرد دست یافت؟
 - چنانچه نیازمندی ای برآورده شد آیا حاصل کار را می توان **آزمود**؟
- مهندس سیستم بایستی از راه فرایند بحث و گفتگو درباره پیشنهادهای مغایر با یکدیگر (مشتری ها) ، راه صحیح و میانه را بیابد. ریسکها همراه با هر نیاز تحلیل می شوند. میزان تاثیر هر نیاز بر هزینه و زمان محاسبه می شود و نیازها به گونه ای درست حذف ، ترکیب یا اصلاح می شوند.

تعیین مشخصات نیازمندیها

يك مشخصه مي تواند يك سند کتبي، الگوي گرافيكي، مدل فرمولي رياضي، مجموعه اي از سناريوهاي مورد استفاده، اشکال اوليه و يا ترکيبي از اين اجزا باشد

«تعیین مشخصات سیستم» مرحله نهایی کار ساخت و نیز کار انجام شده توسط سیستم و مهندس نیازها می باشد. تعیین مشخصات شالوده و زیربنای مهندسی سخت افزار، نرم افزار، مهندسی پایگاه داده و مهندسی انسانی را تشکیل می دهد. عمل تعیین مشخصه ها، حوزه عمل هر یک از ارکان سیستم کامپیوتر را به حدود معینی محدود می نماید. «تعیین مشخصات سیستم» همچنین اطلاعات را که عبارت از ورودی و خروجی های سیستم است تعریف می کند.

مدل سازی سیستم

به منظور تعیین خصوصیات و مشخصات آشپزخانه ای که قرار است ساخته شود، لازم است مدلی معنادار داشته باشیم، یعنی برنامه کار و یا نمایش سه بعدی موجود باشد که موقعیت کابینت ها و لوازم آشپزخانه و هم ارتباط آنها یا یکدیگر را نشان دهد. مدل سازی سیستم نیز به دلیل استدلالهای انجام شده در مورد تهیه برنامه کار یا نمایش سه بعدی آشپزخانه تهیه می شود.

اعتبارسنجی (ارزیابی) نیازمندیها

نتایج مهندسی نیازها در مرحله ارزیابی مورد سنجش قرار می گیرد. به هنگام «اعتبارسنجی» مشخصات سیستم را بررسی می کنند تا مطمئن شوند تمامی نیازمندیها بطور روشن بیان شده، ناهماهنگیها، خطاها و ... کشف و برطرف گردیده و محصول با استانداردهای موجود در روند کار پروژه مطابقت دارد.

اولین گام در ارزیابی نیازمندیها روش بررسی فنی رسمی است که با تشکیل گروهی به بررسی مشخصات سیستم، جهت یافتن خطاهای مربوط به محتوای مشخصات، اطلاعات حذف شده و عدم تطابق ها (تناقض در نیازها و دست یافتنی نبودن آنها) بهترین حالت برای بررسی نیازمندیها استفاده از يك چك لیست (سوال / جواب) است

پرسشهایی در مورد اعتبارسنجی محصول:

- آیا نیازمندیها به وضوح بیان شده است؟ آیا امکان سوء تعبیر آنها وجود دارد؟
- آیا منشا بروز نیازمندی شناخته شده است؟
- آیا آنچه اظهار نهایی نیازمندی مورد بحث استنباط می گردد با خود نیاز مطابقت دارد؟
- آیا در بیان نیازها مقادیر کمی نیز بکار رفته است؟
- چه نیازمندیهای دیگری به نیازمندیهای مطرح شده وابسته است؟
- آیا نیازهای مطرح شده قابل آزمون هستند؟
- آیا پاسخ به نیازمندیها مذکور را می توان در یکی از الگوهای ساخته شده سیستم جست و جو کرد؟
- آیا وجود آن نیازمندیها را می توان میان اهداف کلی سیستم/ محصول جست و جو کرد؟
- آیا نیازمندیها در پیوند با اجزاء سیستم، رفتار و ویژگی عملکردی آن به روشنی شرح داده شده اند؟

مدیریت نیازمندیها

«مدیریت نیازمندیها» عبارت است رشته فعالیتهایی که به گروه امکان می دهد تا ضمن پیشبرد پروژه، در هر مقطع زمانی بتواند نیازها و تغییر نیازها را شناسایی، کنترل و ردیابی نماید.

مدیریت نیازمندیها مانند SCM با عملیات شناسایی آغاز می گردد. و به هر نیاز يك شناسه منحصر به فرد داده می شود که به این صورت است

<شماره نیازمندی> <نوع نیازمندی>

نیاز عملکردی(تابعی): F نیاز داده ها: D نیازمندی رفتاری: B نیازمندی رابط و واسط: I نیازمندی خروجی: P
یعنی نیاز F9 ، نهمین نیاز و از نوع تابعی است

جدولهای ردیابی

• این جدولها ، هر يك ، نیازهای مشخص شده را با يك يا چند جنبه از سیستم با محیط مرتبط می کنند

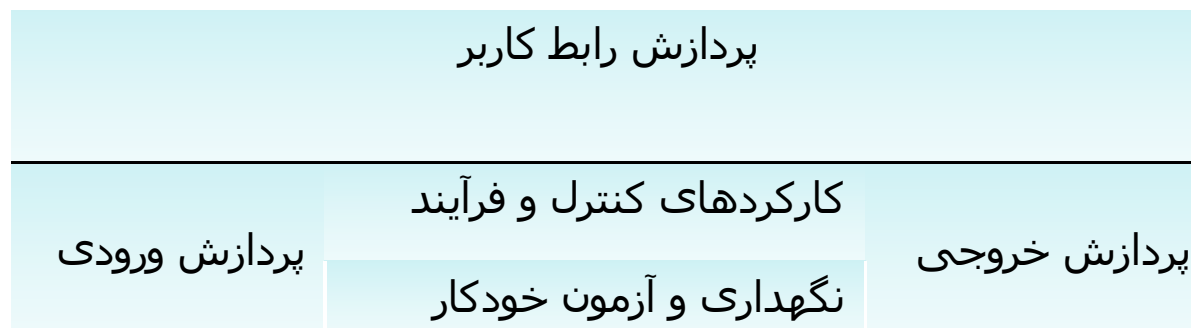
• نیازمندی	01	02	03	04	05	• جنبه ای مشخص از سیستم و محیط			
R_{01}									
R_{02}									
R_{03}									
R_{04}									
R_{05}									
...									
R_{nn}									

انواع جدولهای ردیابی

- «جدول ردیابی ویژگی ها» نشان می دهد که نیازمندیها چگونه با موارد استفاده ویژگیهای قابل توجه سیستم/محصول مرتبطند
- «جدول ردیابی منابع» که منبع هر نیازمندی را شناسایی می کند.
- «جدول ردیابی وابستگی» نشان می دهد که چگونه نیازمندیها به یکدیگر وابستگی دارند.
- «جدول ردیابی زیرمجموعه های سیستم» نیازمندیهای زیرسیستمها را طبقه بندی می کند.
- «جدول ردیابی رابط» نشان می دهد نیازهای چگونه با رابطهای داخلی و خارجی سیستم مرتبط است.

مدل سازی سیستم

- هر سیستم مبتنی بر کامپیوتر را می توان بعنوان سیستم انتقال دهنده اطلاعات که قالب «اطلاعات ورودی-انجام پردازش- اطلاعات خروجی» را بکار می گیرد مدل سازی کرد.
 - برای تهیه و تکمیل الگوی سیستم «قالب مدل سیستم» (الگویی از مدل سیستم) مورد استفاده قرار می گیرد. مهندس سیستم ارکان آن را به یکی از پنج ناحیه پردازشی زیراختصاص می دهد:
- ۱- رابط کاربر
 - ۲- اطلاعات ورودی
 - ۳- عملکرد و کنترل سیستم
 - ۴- اطلاعات خروجی
 - ۵- پشتیبانی و انجام خودآزمون
- الگویی و قالب مدل سیستم در شکل زیر ترسیم شده است



• نمودار بافت (زمینه) سیستم (SCD) System context diagram

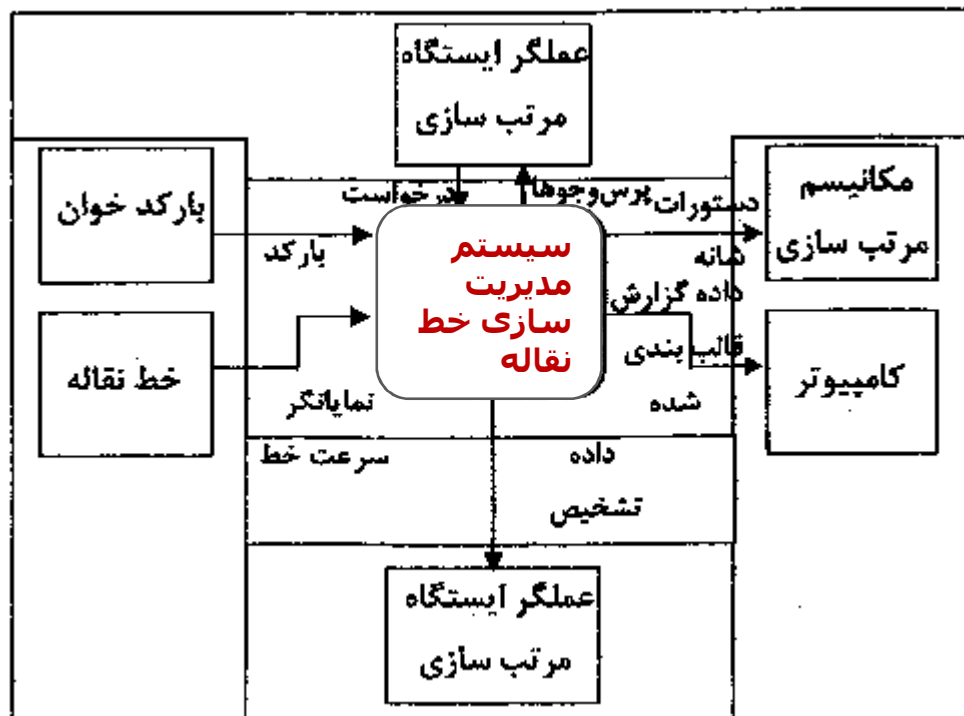
- در راس این سلسله مراتب قرار دارد . این نمودار ، محدوده و مرز اطلاعات **بین سیستم** در حال ساخت و **محیطی** که سیستم مذکور در آن عمل می کند را مشخص می کند. یعنی SCD تعریف کننده تمامی رویه های خارجی اطلاعات مورد استفاده سیستم و ارتباط همه مراجعی که از راه رابط یا انجام پشتیبانی یا خود آزمایی را برقرار می سازد

• مثال نمودار بافت (SCD) در سیستم مرتب سازی (Sorting) (CLSS)

• اهداف CLSS

CLSS باید چنان ارتقاء یابد تا جعبه هایی را که درطول خط انتقال در حرکتند شناسایی کرده و در انتهای خط آنها را بصورت شش «بین» (Bin) مرتب و دسته بندی نماید. جعبه ها توسط ایستگاهی شناسایی، عبور داده می شوند و شماره ای بر روی یک طرف جعبه نقش می گردد. سپس بسوی بین های مربوطه هدایت می شوند. جعبه ها بصورت تصادفی حرکت کرد و بطور برابر جای داده می شوند.

در این مورد CLSS گسترش یافته و از کامپیوتر شخصی در سایت ایستگاه مرتب سازی بهره می گیرد. کامپیوتر شخصی مذکور تمامی اعمال CLSS نرم افزار مربوطه را انجام می دهد.



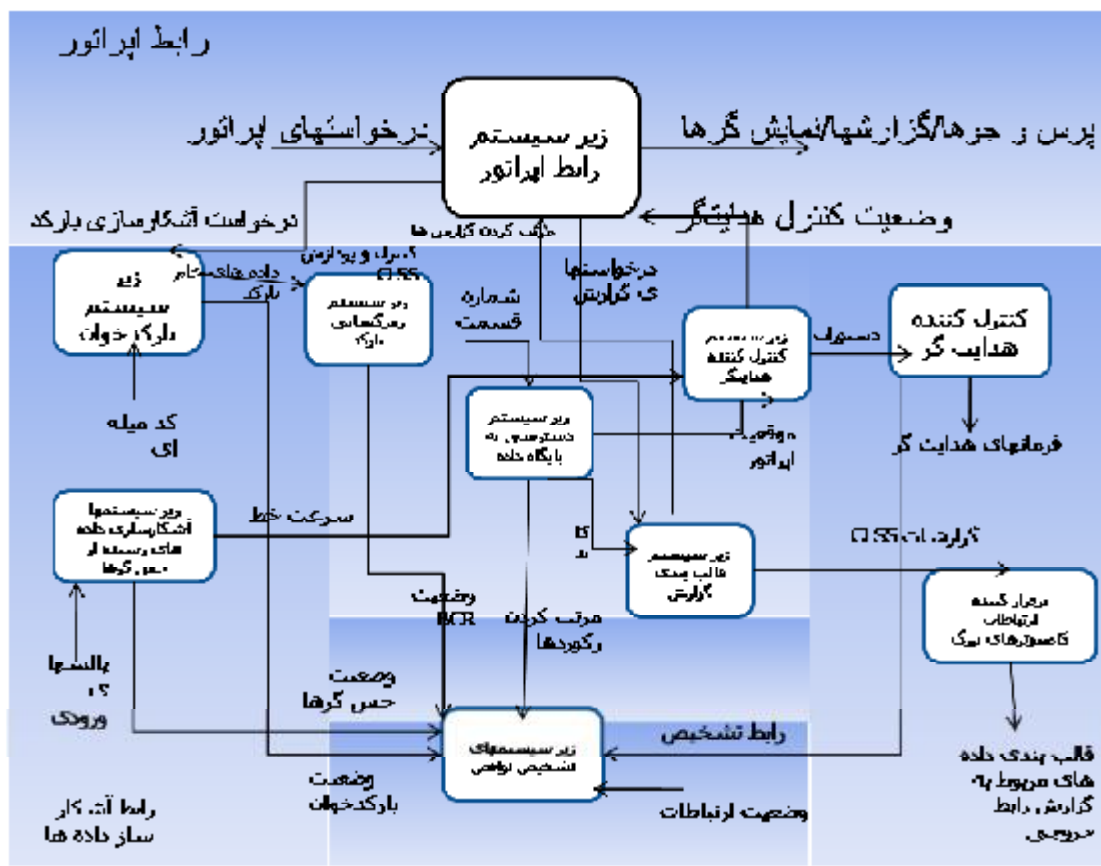
- چهارگوش: نمایانگر یک موجودیت یعنی تولیدکننده یا مصرف کننده
- مستطیل گوشه گرد شده: نماد کل سیستم. از این رو CLSS در ناحیه فرآیند و کنترل در SCD نشان داده شده است.
- پیکان ها: نشانگر اطلاعات هستند از محیط خارجی به درون سیستم CLSS حرکت می کنند.

• نمودار جریان سیستم (SFD) *System Flow diagram*

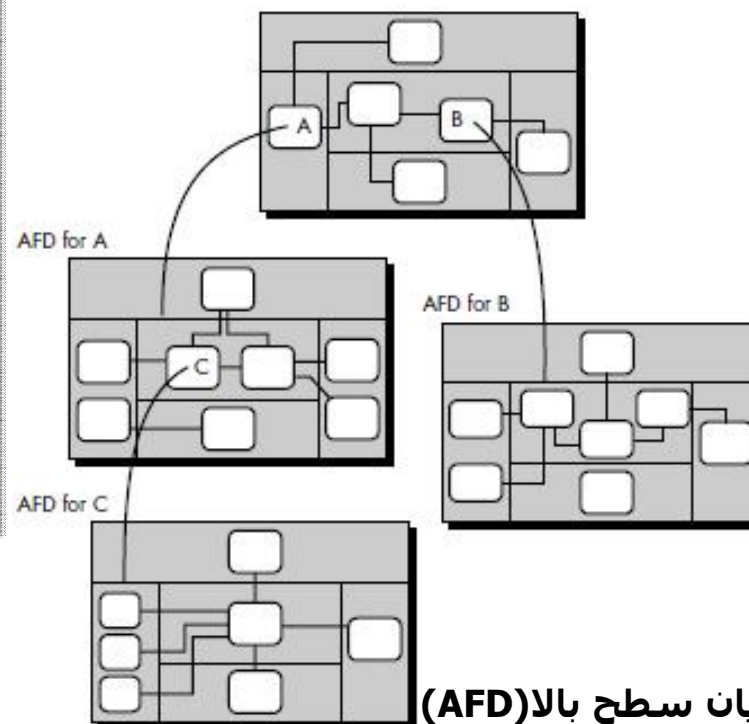
زیر سیستمها در این نمودار تعریف می شوند ، جریان اطلاعات (داده و کنترل) با جزئیات بیشتر ایجاد شده اند.

• زیرمجموعه عمده سیستم در «نمودار جریان سیستم» (SFD) تعریف می شوند که برگرفته از SCD است. اطلاعاتی که در منطقه های SCD جریان دارد مهندس سیستم را در ارتقاء دادن SFD-نمودار CLSS که دقیق تر بوده و جزئیات بیشتری را داراست- راهنمایی و یاری می نماید.

در SFD اصلی هر کدام از چهار ضلعیها با گوشه های گردشده می تواند به الگوی معماری دیگری توسعه یابد.



▪ نمودار جریان سیستم ، زیرسیستم‌های اصلی و خطوط جریانه‌های اطلاعاتی مهم (کنترل و داده) را نمایش می دهد .



• نمودار جریان سطح بالا (AFD)

فصل یازدهم: اصول و مفاهیم تحلیل

□ چه کسی کار را انجام میدهد:

برای بکارگیری تحلیل نیازمندیها در زمینه کارهای تجاری پیچیده یک " **تحلیل گر سیستم**" آموزش دیده در جنبه های تجاری حوزه بکارگیری میتواند این کار را انجام دهد.

□ چرا این کار اهمیت دارد؟

نتیجه تحلیل نکردن این است که پول و زمان هدر رفته و دلسردی شخصی پیش آمده و مشتریان ناراضی خواهند بود.

□ مراحل کار چه میباشد؟

نیازمندیهای داده ها، کارکردی و رفتاری از طریق استخراج اطلاعات از مشتری مشخص میشوند. نیازها پالایش شده و تحلیل میشوند تا وضوح و بی عیبی و سازگاری درونی آنها ارزیابی شود. یک روش تعیین مشخصات که مدلی از نرم افزار را در خود لحاظ کرده است ایجاد شده و سپس توسط مهندسین نرم افزار و مشتریان/کاربران هر دو اعتبارات میابد.

□ حاصل کار چیست؟

یک **طریقه ارائه موثر نرم افزار** باید بعنوان پیامد تحلیل نیازمندیها بوجود آید. نیازمندیهای نرم افزاری مانند نیازمندیهای سیستم میتوانند با استفاده از نمونه اولیه یک مشخص سازی یا حتی مدل نمادین ارائه شوند.

مهندسی نیازهای نرم افزاری یک فرایند کشف، پالایش، مدل سازی و مشخص سازی است.

□ دیدگاه "دونالد ریفر" از مهندسی نیازمندیها:

مهندسی نیازمندیها بکارگیری نظام منداصول اصلاح شده و تکنیک ها و زبانها و ابزارهای تحلیل هزینه موثر و تنظیم سند و تکامل جاری نیازهای مصرف کننده و مشخص ساختن رفتار برونی یک سیستم جهت جلب رضایت ان گروه از نیازهای کاربر میباشد.



تحلیل نیازها

تحلیل نیازمندیها وظیفه مهندسی نرم افزار بوده که شکاف بین مهندسی نیازمندیهای سطح سیستم و طراحی نرم افزار را پر میکند. فعالیت مهندسی نیازمندیها منجر به مشخص سازی خصوصیات عملیاتی نرم افزار شده و بیانگر تعامل با دیگر عناصر سیستم بوده و محدودیتهای نرم افزار را تعیین میکند.

اقسام تحلیل نیازمندیهای نرم افزاری:

۱- فهم مشکل. ۲- ارزیابی و تلفیق. ۳- مدل سازی. ۴- مشخص سازی. ۵- مرور

ابتدا تحلیل گر مشخصات سیستم را مطالعه میکند. و بعد طرح پروژه نرم افزار را بررسی میکند. سپس برای تحلیل باید ارتباطات برقرار شود تا از فهم مشکل اطمینان حاصل شود.

هدف : شناخت پیدا کردن از عناصر مشکل اصلی است انگونه که توسط مشتری/ کاربر دریافت و استنباط میشود.

در طی ارزیابی و یافتن راه حل تمرکز اصلی تحلیل گر بر روی "چه" تا "چگونه" است. (جزئیات توابع نرم افزاری و ...)

سیستم چه داده هایی را تولید و مصرف میکند؟ چه کارکرد هایی تعریف شده اند؟ و چه محدودیت هایی اعمال میشوند؟

در طی دوره ارزیابی و فعالیت یافتن راه حل تحلیل گر مدل هایی از سیستم در تلاش برای درک بهتر داده ها و جریان کنترل، پردازش کارکردها، رفتار عملیاتی و مضمون اطلاعات را خلق میکند. این مدل بعنوان زیر بنای طراحی نرم افزار و پایه خلق یک مشکل سازی برای نرم افزار خدمت خواهد کرد.

تعیین نیازمندیهای نرم افزار:

قبل از آنکه بتوان نیازمندیها را تحلیل و مدل سازی و یا مشخص نمود باید آنها را طی یک پروسه استخراج گردآوری نمود.

□ راه اندازی فرایند:

رایج ترین تکنیک استخراج نیازمندیها برگزاری نشست با مصاحبه میباشد. بنابراین ارتباط باید شروع شود.

" گاز و وینبرگ " : پیشنهاد میدهند که تحلیل گر ملاقات را با طرح سوالات خارج از مضمون اصلی شروع کند. یعنی یک سری سوالات که به شناخت اساسی مشکل افرادی که برای آن راه حل دارند و ماهیت راه حل مطلوب و کارایی اولین دیدار منتهی خواهد شد.

اولین سری سوالات رها از مضمون اصلی بر روی مشتری و اهداف و منابع کلی متمرکز میشوند. مثلاً:

۱- چه کسی تقاضای این کار را داشته؟

۲- چه کسی از این راه حل استفاده خواهد کرد؟

۳- سود اقتصادی یک راه حل موفق چه خواهد بود؟

۴- آیا منبع دیگری برای راه حل مورد نظر شما وجود دارد؟

سری بعدی سوالات تحلیل گر را قادر میکند تا درک بهتری از مشکل و مشتری پیدا کرده و نظرات خود را درباره راه حل به او بگوید:

۱- چگونه خروجی خوب را که یک راه حل موفق دنبال می آورد تشریح میکنید؟

۲- این راه حل چه مشکلاتی را مورد توجه قرار میدهد؟

۳- آیا امکان دارد که محیط بکار گیری راه حل را برایم تشریح کنید؟

۴- آیا مباحث ویژه عملکردی یا محدودیت هایی در مسیر یافتن راه حل وجود دارد؟

آخرین سری سوالات بر روی کارایی جلسه متمرکزند. **(فرا پرسش)**

" گایز و وینبرگ " به آنها **فوق سوالات** گفته و لیست مختصر شده زیر را پیشنهاد میدهند:

۱- آیا شما فرد مناسبی برای پاسخ گویی به این سوالات هستید؟

۲- آیا پاسخ های شما رسمی هستند؟

۳- آیا سوالات من به مشکل شما مربوط میشوند؟

۴- آیا من بیش از اندازه در حال طرح سوالات هستم؟

۵- آیا فرد دیگری وجود دارد که بتواند اطلاعات اضافی در اختیار بگذارد؟

۶- آیا مورد دیگری وجود دارد که از شما سوال کنم؟

این سوالات به " شکستن یخ " کمک کرده و ارتباط را برقرار میکند که برای موفقیت تحلیل گر ضروری میباشد.

جلسه پرسش و پاسخ فقط برای اولین برخورد است و سپس باید جلساتی با چارچوب تلفیق عناصر حل مشکل مذاکره و مشخص سازی همراه آن برگزار شود.



فنون تسهیل مشخص سازی کاربردی: FAST

غالباً مشتریان و مهندسین نرم افزار دارای یک ذهنیت ناخود آگاه " ما " و " انها " هستند. (بجای آنکه بصورت تیم فعالیت کنند) که این رهیافت نتیجه خوبی به همراه ندارد. سوء تفاهم در حد بالا و اطلاعات مهم حذف شده و نیز هیچگاه رابطه کاری موفق برقرار نمیشود.

روش فنون تسهیل مشخص سازی (**FAST**) ایجاد یک تیم مشترک از مشتریان و تحلیل گران را تشویق میکند تا با یکدیگر در زمینه **تعیین مشکل**. پیشنهاد عناصر **راه حل**. مذاکره در مورد راهکارهای مختلف و مشخص ساختن یک مجموعه ابتدایی نیازمندیهای راه حل همکاری نمایند.

این تکنیک پتانسیل بهبود روابط را در کلیه زمینه های کاربردی ارائه میکند.

انواع رهیافت های FAST تغییراتی را اعمال میکنند که رهنمودهای زیر انها را ارائه میکند:

۱- ملاقاتی در مکانی بی طرف برگزار و مهندس نرم افزار و مشتریان حضور دارند.

۲- قوانین برای آماده سازی و تدارک تثبیت شده اند.

۳- دستور کاری پیشنهاد میشود که انقدر رسمی است تا نکات مهم را پوشش دهد ولی انقدر غیر رسمی نیز هست تا جریان آزاد عقاید را امکانپذیر می سازد.

۴- یک "تسهیل گر" جلسه را کنترل میکند. (یه نفر جلسه را کنترل کند)

۵- یک "مکانیسم تعریف" بکار میرود. (تابلوی اعلانات؛ نمودار خاص..)

۶- هدف تعیین مشکل. پیشنهاد عناصر **راه حل**. مذاکره در مورد رهیافتهای گوناگون و تعیین مجموعه مقدماتی نیازمندیهای راه حل در یک فضای مفید برای تحقق هدف میباشد.

ملاقاتهای اولیه بین مشتری و مهندس نرم افزار برگزار و پرسش و پاسخهای اساسی به روشن شدن دامنه مشکل کمک کرده و ادراک کلی از راه حل بدست می آید. ماحصل این جلسات اولیه آن است که مشتری و مهندس نرم افزار هر دو یک گزارش یک یا دو صفحه ای درباره درخواست محصول مینویسند. (نتیجه : جلسات با سه دسته سوال)

مکان. زمان و تاریخ جلسه **FAST** انتخاب شده و یک تسهیل گر (ناظر) نیز انتخاب می شود.

شرکت کنندگان از هر دو طرف مشتری و مهندس نرم افزار برای شرکت در جلسه دعوت شده و درخواست محصول قبل از برگزاری جلسات توزیع می شوند.

از هر شرکت کننده خواسته می شود که (۱) لیستی از اشیاء، موضوعات اصلی، خدمات، قیود و معیارهای عملکردی تهیه نمایند. لیست نباید خسته کننده باشد اما باید چشم انداز خود را درباره سیستم منعکس کند. حتی با ارائه اطلاعات بیشتر هنوز ابهاماتی وجود دارد. احتمال حذف مواردی و امکان وقوع خطا نیز وجود دارد.

تیم "فنون تسهیل مشخص سازی" متشکل از نمایندگان از بخش های بازاریابی و مهندسی نرم افزار و سخت افزار و تولید می باشد. یک تسهیل گر (ناظر) خارجی نیز باید حضور داشته باشد.

هر شرکت کننده لیستهای عوامل محدود کننده خود را تهیه میکند. (۲) پس از شروع جلسه اولین موضوع بحث نیاز و توجیه محصول جدید است. لیستها باید به نحوی ارائه شوند که بتوان آنها را با یکدیگر ترکیب کرده یا حذف کرده و یا اضافاتی در آنها انجام داد. در این مرحله انتقاد و بحث اکیدا ممنوع است.

پس از ارائه لیستها جداگانه (۳) یک لیست ترکیب توسط گروه در مورد یک موضوع تهیه میشود که این لیست لیستهای تکراری را حذف کرده و نظرات جدید را به آنها اضافی میکند اما چیزی را پاک نمیکند.

پس از گردآوری و تنظیم لیستهای ترکیب (۴) میباحثه که توسط تسهیل گر هماهنگ می شود آغاز می گردد. لیست ترکیب کوتاه یا بلند شده یا مجددا نوشته می شود تا بتواند بخوبی منعکس کننده سیستم/محصول پیشنهادی برای تولید باشد.

هدف : بدست آوردن یک لیست "اتفاق نظر" در هر زمینه و موضوع می باشد.

زمانیکه لیستهای اتفاق نظر تکمیل شدند (۵) تیم مربوطه به تیم های کوچک تری تقسیم میشود که هر یک در جهت تهیه یک مشخص سازی کوچک برای یک یا چند مورد اضافه شده به هر یک از لیست ها فعالیت خواهد کرد.

(۶) مینی - مشخص سازی "(جزئیات) پرداختن معضل به یک کلمه یا عبارت بکار رفته در یک لیست می باشد.

هر تیم فرعی و سپس هر یک از مینی - مشخص سازیها خود را به کلیه شرکت کنندگان در FAST برای بحث روی آن ارائه می کند. اضافات و موارد حذف و توضیحات بیشتر انجام میشود. در برخی موارد توسعه مینی - مشخص سازیها به اشیاء خدمات یا محدودیتهای نیازمندیهای عملکردی جدید منتهی میشود که به لیستهای اصلی اضافه خواهند شد.

پس از آنکه مینی - مشخص سازیها تکمیل شد هر شرکت کننده در جلسه یک (۷) لیست معیار اعتبار برای محصول/سیستم تهیه کرده و لیست خود را به تیم ارائه میدهد. سرانجام یک یا چند شرکت کننده (۸) پیش نویس کامل مشخصات را بر عهده می گیرد.



تنظیم کارکرد کیفیت:

Quality Function Deployment

گسترش (تنظیم) کارکرد کیفیت QFD تکنیک مدیریت کیفیتی است که نیازهای مشتری را به نیازهای فنی نرم افزار ترجمه می کند.

QFD سه نوع نیازمندی را شناسایی کرده است:

- ۱- نیازمندیهای عادی : مقاصد و اهدافی که برای محصول یا سیستم در طول ملاقات با مشتری مطرح شده اند . چنانچه این نیازمندیها برآورده شوند انگاه مشتری راضی خواهد بود.
 - ۲- نیازمندیهای مورد انتظار : این نیازمندیها میتوانند به صورت تلویحی (ضمنی) درباره محصول یا سیستم مطرح شده و به میزانی بنیادی باشند که مشتری آنها را بطور صریح مطرح نکند. فقدان آنها میتواند باعث نارضایتی شدید شود.
 - ۳- نیازمندیهای هیجان انگیز : این ویژگی ها فراتر از انتظار مشتری رفته و زمانی که ارائه می شوند بسیار رضایت بخش خواهند بود.
- QFD** از مصاحبه ها، مشاهدات، پژوهش ها و آزمون های تاریخی داده ها بعنوان داده های خام برای جمع اوری نیازها استفاده می کند. سپس این داده ها به جدول نیازها - که جدول صدای مشتری نامیده میشود - ترجمه می شوند که از سئای مشتری مورد بررسی قرار می گیرند. انواع دیاگرام ها و پانویس ها و روش های ارزشیابی برای استخراج نیازهای مورد انتظار بکار می روند و تلاش می شود تا نیازهای هیجان انگیز استخراج شوند.

USE-CASE:

مهندس نرم افزار (تحلیل گر) میتوانند مجموعه ای از سناریوهایی را که یک ریسمان کاربرد (مسیر استفاده) سیستمی که باید ساخته شود را تعیین نمایند. این سناریوها که غالب **USE-CASE** نامیده می شوند شرحی از **چگونگی استفاده از سیستم** را فراهم می کنند. برای ایجاد یک **USE-CASE** باید در **آغاز انواع مختلف افراد/ابزارها را که از سیستم یا محصول استفاده خواهند کرد تعیین نماید**. این **بازیگرها** نقشهایی را بعنوان **اپراتورهای سیستم** ایفا میکنند را نشان میدهند. یک بازیگر زمانی که بطور رسمی تر تعیین میشود همان چیزی است که با سیستم با محصول ارتباط برقرار میکند و نسبت به سیستم یک **عامل خارجی** محسوب می شود. هر بازیگری و یا کاربری با یکدیگر یکسان نیستند . یک کاربر معمول می تواند تعداد مختلفی نقش را بهنگام بکارگیری سیستم ایفا نماید . در حالیکه **بازیگر** نماینده گروهی از عناصر خارجی می باشد که فقط ایفاگر یک نقش هستند.

از آنجا که استخراج نیازمندیهای یک فعالیت رو به تکامل است تمام بازیگرها در ظرف اولین تکرار مشخص نمی شوند. امکان دارد که بازیگران اولیه را در اولین تکرار تعیین کرده و بازیگران ثانویه را پس از شناخت بیشتر سیستم معین کرد. بازیگران اولیه برای دستیابی به کارکرد مطلوب سیستم وارد تعامل میشوند و سود مورد نظر از سیستم را حاصل میکند. آنها مستقیماً و اغلب با نرم افزار کار میکنند.

بازیگران ثانویه برای حمایت سیستم حضور دارند با بازیگران اولیه بتوانند در کار خود موفق شوند. "یاکوبسن" پیشنهاد میدهد که **سوالات** زیر برای **توسعه USE-CASE** پاسخ داده شوند:

- ۱- وظایف و کارکرد های اصلی که بازیگر باید انجام دهد چیست؟
 - ۲- چگونه سیستم اطلاعات توسط بازیگر تولید تغییر یافته یا بدست می آید؟
 - ۳- آیا بازیگر باید سیستم را در مورد تغییرات در محیط خارجی آن مطلع کند؟
 - ۴- چه اطلاعاتی را بازیگر از سیستم میطلبد؟
 - ۵- آیا بازیگر علاقمند به مطلع شدن درباره تغییرات غیر منتظره هست؟
- بطور کلی یک **USE-CASE** یک شرح روایتی ساده است که نقش هر بازیگر را ضمن تعامل با سیستم روشن میکند.
- USE-CASE** ها سناریوهایی را شرح میدهند که توسط بازیگرهای مختلف بصورت متفاوت فهمیده میشوند.
- "وایدر" پیشنهاد میکند:

پیاده سازی کارکرد کیفیت میتواند برای توسعه یک مقدار اولویت ارزیابی شده در رابطه با هر UseCase به کار گرفته شود. برای تحقق این امر **USE-CASE** از نقطه نظر تمام بازیگران تعریف برای سیستم ارزیابی می شوند. یک مقدار اولویت توسط هر بازیگر برای هر **USE-CASE** تعیین میشود.

سپس یک اولویت میانگین محاسبه شده و اهمیت ادراک شده هر یک از **USE-CASE** مشخص میشود.

زمانی که یک مدل فرایند تکرار برای مهندسی نرم افزار معطوف به اشیاء بکار میرود اولویتها می توانند بر روی تعیین اولین قابلیت کارکردی سیستم که می تواند ارائه شود تاثیر بگذارند.



اصول تحلیل:

پژوهشگران مشکلات تحلیل را تعیین کرده و با تشخیص دلایل بوجود آمده آنها انواع راه های مدل سازی و مجموعه های راه گشای موثر در برطرف کردن آنها را ارائه داده اند.

هر مدل تحلیل دارای یک نکته منحصر بفرد است .

تمام مدل های تحلیل توسط مجموعه ای از اصول عملیاتی با یکدیگر مرتبطند (موارد مشابه در مدل های تحلیل)

- ۱- حوزه اطلاعات یک مرحله باید بازنمایی و ادراک شود.
 - ۲- کارکردهایی که نرم افزار باید انجام دهد مشخص شوند.
 - ۳- رفتار نرم افزار باید بازنمایی شود.
 - ۴- مدل هایی که نمایشگر اطلاعات و کارکرد و رفتار که باید به روشی تقسیم بندی شوند که جزئیات را به طریقی لایه بندی شده (سلسله مراتبی) اشکار نماید.
 - ۵- فرایند تحلیل باید از اطلاعات اساسی به سوی جزئیات پیاده سازی حرکت کند.
- “ دیویس ” یک مجموعه از اصول راهنما را برای مهندسی نیازها پیشنهاد میکند:**

- ۱- مشکل را قبل از شروع به ایجاد مدل تحلیل درک کنید.
- ۲- نمونه های اولیه را ایجاد کنید که کاربر را قادر می سازد تا چگونگی تعامل کاربر- ماشین را ادراک نماید.
- ۳- ریشه و دلیل هر نیازمندی را ثبت کنید.
- ۴- از نظرات مختلف درباره نیازمندیها استفاده کنید.
- ۵- طبقه بندی نیازمندی ها ، موعدهای زمانی می توانند پیاده سازی هر نیازمندی نرم افزاری را متوقف کنند.
- ۶- بر روی رفع ابهامات کار کنید.



میدان اطلاعات:

تمام کاربردها نرم افزار را میتوان بصورت گروهی "پردازش داده ها" نامید.

نرم افزار برای پردازش داده ها تغییر آنها از شکلی به شکل دیگر بوجود می آید. عبارت دیگر پذیرش ورودی به معنی کنترل و پردازش آن و سپس بیرون دادن آن است.

باید دقت کنیم نرم افزار رخدادهای را نیز پردازش میکند. یک واقعه نماینده جنبه ای از کنترل سیستم است که در واقع چیزی بیش از داده های صفر و یک نیست.

داده ها (اعداد؛ متن، تصاویر و...) و کنترل (وقایع) هر دو در میدان اطلاعات یک مسئله قرار می گیرند. اولین اصل تحلیل عملیاتی نیازمند یک میدان اطلاعات و ایجاد یک مدل داده ها میباشد.

میدان اطلاعات در بر دارنده سه نقطه نظر در رابطه با داده ها و کنترل است:

۱- مضمون (محتوای) اطلاعات و روابط (مدل داده ها).

۲- جریان اطلاعات.

۳- ساختار اطلاعات برای درک کامل میدان اطلاعات.

مضمون اطلاعات نماینده داده های جداگانه و اشیاء کنترل است که مجموعه ای بزرگتر از اطلاعات تغییر یافته توسط نرم افزار را تشکیل میدهند.

اشیاء داده ای و کنترل را میتوان به اشیاء داده ای و کنترل دیگر ربط داد.

جریان اطلاعات: نماینده طریقه تغییر داده ها و کنترل چگونه ای است که هر یک در سیستم حرکت می کنند. مطابق شکل (ص ۳۶۸) اشیاء ورودی به اطلاعات میانی تغییر شکل میدهند که در نهایت به خروجی تبدیل می شوند.

در راستای این تغییر شکل مسیر اطلاعات اضافی می توانند از داده های موجود ذخیره شده معرفی شوند.

ساختار اطلاعات : نماینده سازمان درونی انواع داده ها و قلم های کنترل است. (ارایه - کلاس یا ...).

مدل سازی

مدلهای کارگردی برای درک بهتر آنچه باید در عمل ساخته شوند خلق می شوند.

زمانی که این مسئله یک مورد فیزیکی است : مدلی ایجاد می کنیم که از نظر فرم و شکل یکسانند ولی از نظر مقیاس کوچکترند. زمانی که مسئله مورد نظر نرم افزار باشد مدل باید نماینده اطلاعاتی باشد که نرم افزار تغییر شکل میدهد. اصول دوم و سوم تحلیل عملیاتی نیازمند ساخت مدلهای کارکرد و رفتار است.

مدلهای کارکردی :

نرم افزار اطلاعات را با حداقل سه کارکرد عمومی تغییر شکل میدهد :

۱- ورودی. ۲- پردازش. ۳- خروجی.

سپس مهندس نرم افزار بر روی کاراکترهای ویژه مسئله متمرکز می شود.

مدل کارکردی با یک مدل در سطح بافت منفرد شروع می شود . طی یک سری تکرارها جزئیات کاراکتری بیشتری فراهم می شود تا اینکه یک تصویر از تمام قابلیت های کارکردی سیستم ارائه شود.

مدل رفتاری :

اکثر نرم افزارها به رخداد از جهان بیرونی پاسخ میدهد. این خصوصیت محرک - پاسخ اساس مدل رفتاری را تشکیل میدهد.

مدلهای فراهم شده در طی تحلیل نیازمندیها چند وظیفه و نقش را بر عهده دارند:

۱- مدل به تحلیل گر کمک میکند تا اطلاعات و کارکرد و رفتار سیستم را درک کرده و از این طریق امر تحلیل نیازها را ساده تر کرده و بیشتر نظام مند نماید.

۲- مدل نقطه کانون توجه می شود و بنابراین کلیدی برای تعین کامل بودن سازگاری و دقت مشخص سازی خواهد بود.

۳- مدل پایه طراحی واقع شده و برای طراح شناختی اساسی از نرم افزار فراهم میکند تا بتواند برای پیاده سازی آن در بافت مربوطه نقشه تهیه نماید.



افراز (تجزیه و تقسیم):

مسائل اغلب بیش از اندازه بزرگ و پیچیده هستند بنابراین ما تمایل داریم تا آنها را **تقسیم بندی** نماییم. تحلیل چهارم عملیاتی چنین نظر میدهد که اطلاعات و کارکرد و حوزه رفتاری نرم افزار تقسیم بندی شوند. تجزیه و تفکیک فرایندی است که نتیجه آن در تهیه کامل داده ها و کارکردها یا رفتار مشهود است . آنرا میتوان بطور افقی یا عمودی انجام داد.

طریقه تقسیم بندی:

۱- عرضه جزئیات رو به افزایش با حرکت **عمودی** در سلسله مراتب.

۲- متلاشی نمودن مسئله بطور **کار کردی** از طریق حرکت **افقی** در سلسله مراتب.

دیدگاه اساسی و پیاده سازی:

یک **دید ضروری** از نیازمندیهای نرم افزار کارکردهایی را که باید اجرا شوند و اطلاعاتی را که باید پردازش شوند را بدون در نظر گرفتن جزئیات پیاده سازی ارائه میکند.

دید پیاده سازی نیازمندیهای نرم افزار ظاهر شدن کارکردهای پردازش در دنیای واقعی و ساختارهای اطلاعات را میسر میکند.

تحلیل گر باید محدودیت های تحمیل شده را که از سوی عناصر سیستم از پیش تعریف شده متوجه شده و دید پیاده سازی کارکرد و اطلاعات را زمانی که چنین دیدی مناسب است در نظر بگیرد.

دید پیاده سازی نباید ضرورتاً بعنوان یک بازنمایی چگونگی تفسیر شود بلکه مدل پیاده سازی نمایندگی حالت عملیاتی جاری را بر عهده دارد.

مدل اساسی را بر این مفهوم که عملی شدن کارکرد بطور صریح اظهار نشده است متداول است.



نمونه سازی نرم افزار:

اشکال تحلیل با یکدیگر متفاوتند .

در برخی موارد امکان اعمال اصول تحلیل عملیاتی و بر گرفتن یک **مدل نرم افزاری** برای کار طراحی امکانپذیر است. در دیگر موقعیت ها استخراج نیازمندیها انجام میشود. اصول تحلیل اعمال میشوند و **مدلی برای نرم افزاری** ایجاد میشوند که **نمونه اولیه نامیده می شود.**

نمونه اولیه برای ارزیابی مشتری و توسعه دهنده ساخته شده است. در اخر شرایطی وجود دارند که ساخت یک نمونه اولیه را برای آغاز تحلیل ضروری میسازند. سپس این مدل بصورت نرم افزار نهایی به تکامل میرسد.

انتخاب رهیافت نمونه سازی:

پارادایم ساخت نمونه اولیه می تواند " انتها - بسته " یا " انتها - گشوده " باشد.

راهکار انتها - بسته فالباً " نمونه دور انداختنی اولیه " نامیده میشود. با بکار گیری این راهکار یک نمونه اولیه فقط بعنوان یک نماد کلی از نیازمندیها بکار می آید.

راهکار انتها - گشوده " نمونه اولیه سازی تکاملی " نامیده میشود. این راهکار از نمونه اولیه بعنوان اولین قسمت کار تحلیلی که به طراحی و تولید منتهی خواهد شد استفاده میکند.

نمونه اولیه نرم افزار اولین مرحله تکاملی سیستم به اتمام رسیده است.

پیش از انتخاب یکی از این دو راهکار باید در نظر گرفت که آیا برای سیستم مورد نظر میتوان نمونه اولیه ساخت یا خیر؟

عوامل کاندیدا در نمونه سازی اولیه:

- ۱- محدوده بکار گیری. ۲- پیچیدگی بکار گیری.
- ۳- خصوصیات مشتری.
- ۴- خصوصیات پروژه

کاندیدا نمونه اولیه سازی:

هر گونه کاربردی که صفحه نمایش های تصویری پویا خلق می کند شدیداً با کاربر تعامل میکند و یا به الگوریتم ها و فرایندهای ترکیبی نیاز دارد که باید بطریق تکاملی بوجود آیند را کاندیدایی برای نمونه اولیه سازی تلقی می کنیم.

اگر بکارگیری یک کاندیدا نیاز به ایجاد ده ها هزار خط برنامه داشته باشد نمونه اولیه سازی بیش از حد پیچیده است. اگر بتوان پیچیدگی را تقسیم بندی کرد امکان ساخت نمونه اولیه برای قسمتهای اولیه نرم افزار وجود دارد.

اگر مشتری بخواهد با نمونه اولیه در مراحل بعدی تعامل کند باید:

۱- منابع مشتری در خدمت ارزیابی و پالایش نمونه اولیه باشد.

۲- مشتری قادر به اتخاذ تصمیمات مربوط به نیازمندی بصورت بهنگام باشد.

شیوه ها و ابزار های نمونه سازی:

برای کارآمد بودن نمونه اولیه نرم افزار یک نمونه اولیه باید با سرعت ایجاد شود تا مشتری نتایج را ارزیابی کرده و تغییرات را توصیه کند.

روشها و ابزارهایی که برای ارائه سریع نمونه اولیه سازی فراهم هستند:

۱- فنون نسل چهارم :

طبقه وسیعی از پرسش ها بر پایه داده ها و گزارش زبانها و برنامه و مولدهای بکارگیر و سایر زبانهای غیر - رویه ای سطح بالا را در بر میگیرد. این فن از آنجا که مهندس نرم افزار را قادر به اجرا سریع برنامه میسازد برای نمونه اولیه سازی ایده آل است.

۲- اجزاء قابل استفاده مجدد نرم افزار :

راهکار دیگر جفت و جور کردن و سوار کردن اجزاء بجای ساختن نمونه اولیه است که با کمک اجزاء نرم افزار های موجود بکار گرفته میشود.

۳- مشخصات رسمی و محیط های نمونه اولیه سازی:

طی دو دهه گذشته تعدادی ابزارهای مشخص سازی رسمی بوجود آمده اند که برای فنون مشخص سازی زبان جانشین می شوند.

- طراحان این زبانهای رسمی در تعامل فرایند ایجاد محیط های تعاملی هستند که :
- ۱- تحلیل گر را قادر سازد تا بطود تعاملی یک مشخص سازی بر پایه زبان از سیستم یا نرم افزار را **خلق** نماید.
 - ۲- از ابزار های خود کاری مدد بگیرد که مشخص سازی بر پایه زبان را برای برنامه های قابل اجرا **ترجمه** کند.
 - ۳- مشتری را قادر سازد تا برنامه قابل اجرا نمونه اولیه را برای **پالایش نیاز مندیهای رسمی** بکار گیرد.

تعیین مشخصات:

حالت **مشخص سازی** با **کیفیت** راه حل ارتباط زیادی دارد.

اصول تعیین مشخصات:

مشخص سازی می تواند بعنوان یک فرایند نماد و نمود تلقی شود. نیازها به طریقه ای ارائه شده اند که در نهایت به پیاده سازی نرم افزار موفق منتهی می شود.

اصول مشخص سازی بر گرفته از تحقیق " بالزر و گودمن " :

۱- جدا کردن قابلیت کارکردی از پیاده سازی.

۲- توسعه مدلی از رفتار مطلوب سیستمی که داده ها و پاسخ های کارکردی سیستم را به انواع محرک های محیط در بر میگیرد.

۳- ایجاد متنی که نرم افزار از طریق مشخص سازی طریقه تعامل اجزاء سیستم با نرم افزار در ان عمل نماید.

۴- تعریف محیطی که سیستم در ان عمل میکند و روشن کردن چگونگی واکنش مجموعه ای شدیداً در هم تنیده از عوامل با محرک درون محیطی که توسط ان عوامل ایجاد شده است.

۵- ایجاد یک مدل عقلانی تا یک مدل طراحی یا پیاده سازی یک سیستم را انگونه که از سوی جامعه کاربران ادراک می شود تشریح میکند.

۶- فهم این مسئله که : مشخص سازی باید نواقص را تحمل کرده و قابل بهبود باشد .

۷- ایجاد مضمون و ساختار مشخص سازی به طریقه ای که انرا قابل تطبیق با تغییرات نماید.

باز نمایی:

نیازمندیهای نرم افزار می تواند به طرق مختلف مشخص شود.

در این مورد رهنمودهای زیر ارزش بکارگیری دارد:

۱- نمایش و ارائه چارچوب و مضمون باید با مسئله مرتبط باشد.

۲- اطلاعات لحاظ شده در مشخص سازی باید تو در تو باشند.

۳- دیاگرامها و دیگر علائم باید از نظر تعداد محدود شده و از نظر بکارگیری سازگاری داشته باشد.

۴- بازنمایی باید قابل تجدید نظر باشد.

تعیین مشخصات نیازمندیهای نرم افزار :

مشخص سازی نرم افزار حاصل نهایی کار تحلیلی است.

معرفی مشخص سازی نیازهای نرم افزار بیانگر اهداف و مقاصد نرم افزار و شرح آن در بافت سیستم استوار بر کامپیوتر است.

شرح اطلاعات یک توصیف تفصیلی از مسئله ای را که نرم افزار باید طی کند فراهم می کند . مضمون اطلاعات و جریان و ساختار آن مستند می شوند. سخت افزار و نرم افزار و تعاملات انسانی برای عناصر سیستم خارجی و کارکردهای درونی نرم افزار شرح داده شده اند.

شرح هر کارکرد در شرح کارکردی ارائه میشود . یک شرح روایتی پردازش گر برای هر کارکرد ارائه میشود. محدودیتهای طراحی بیان و توجیه میشوند . خصوصیات عملکردی آن اظهار شده و یک یا چند دیاگرام برای نمایش گرافیکی ساختار کلی نرم افزار لحاظ میشوند.

شرح رفتاری مشخص سازی عملیات نرم افزار را بعنوان خصوصیات سلسله ای از حوادث بیرونی و درونی مورد ازمون قرار میدهد.

معیار اعتبار سنجی مهمترین و در عین حال مورد بی اعتنائی ترین بخش مشخص سازی نیازهای نرم افزار است.

سرانجام مشخص سازی شامل یک " کتاب شناسی " و " پیوست " میشود.

کتاب شناسی ارجاع به تمام مستندات مربوط به نرم افزار دارد.



باز بینی مشخصات:

از آنجا که مشخص سازی زیر بنای مرحله توسعه را تشکیل داده باید در مورد بررسی و مرور دقت زیادی صورت گیرد. مرور در آغاز در سطح ماکروسکوپی (کلان) انجام میشود. برای بررسی کامل هر یک از حوزه ها باید بررسی دارای جزئیات بیشتری شود و نیز طریقه ارائه شدن نیازمندیها بررسی و کنترل می شود.

پس از کامل شدن بررسی مشخص سازی نیازهای نرم افزار از سوی تحلیل گر و مشتری به کناری نهاده می شود. مشخص سازی برای توسعه نرم افزار مانند یک "قرارداد" شده است. تقاضا در مورد اعمال تغییرات در نیازها پس از نهای شدن مشخص سازی حذف نخواهند شد. اما مشتری باید بداند که اینگونه تغییرات پس از نهای شدن کار بمعنای صرف هزینه بیشتر قرارداد خواهد بود.

ازموند مشخص سازی بهر روش معنی داری مشکل و دشوار است. بنابراین عدم سازگاری یا موارد حذف شده می توانند بدون آنکه تشخیص داده شوند وارد کار شوند. در طول بررسی تغییرات اعمال شده در مشخص سازی توصیه می شود

این جزوه خلاصه بوده و سعی شده مطالب اصلی و مهم درج شود و موارد اشتباه و نادرست حذف شده باشد
امیدوارم توانسته باشم کمکی به همه دوستان و دانشجویان پیام نوری و اساتید و همکارانی که در این ترم از
اسلایدها استقبال نمودند و به عنوان کمک آموزشی استفاده کردند و سایر افرادی که نیاز به مطالعه کتاب
دارند انجام داده باشم.....اشتباهات موجود را به بزرگواری خود ببخشین.
شاد و پیروز باشین - یا حق - ایمان مختاری