مهندسی نرم افزار

مقدمه :

یک مهندس کامپیوتر موفق کسی است که تمام نرم افزارهای مورد نیاز خود را به راحتی در دسترس داشته و کار با آنها برایش ساده باشد . در حقیقت کسی که بتواند با نرم افزارهای کامپیوتری خوب کار کند مهندس کاملی است امروزه بسیاری از شرکتها نرم افزارهای کامپیوتری خود را به صورت آشفته و در هم تولید می کند که گرچه help آنها در رهیافت تخصصی در رفع عیب آنها دخیل است اما هنوز شرکتی که در ارائه رفع عیب المانهای سیستم کارکرد جوی داشته باشد وجود ندارد وب سایت http:mhhd.com برای منابع مهندسی نرم افزار بسیار مفید است .

ما در این جزوه سعی می کنیم محصول و فرآیند نرم افزاری را مورد بحث قرار داده سپس به طرح ریزی و مدیریت آن بپردازیم و روش تحلیل را بررسی کرده و به سیستمهای شی گرا و مقایسه آن با دیگر سیستمها بپردازیم مبحث های Dhtml , Uml, GQM و سایر مطالب نرم افزاری در این بحث مورد مطالعه قرار می گیرد .

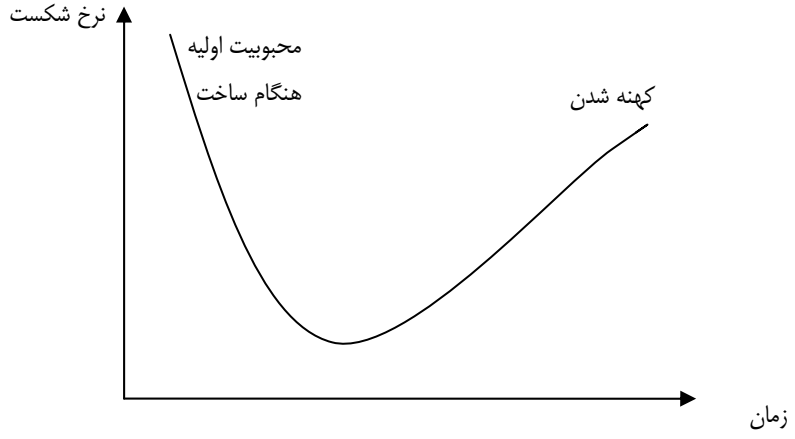
نکته : سایت www.Sepa.com سایت مفیدی برای این درس است .

یک نرم افزار کامپیوتری چیست ؟

محصولی که یک مهندس نرم افزار آن را تعریف می کند (حین طراحی) و به تولید آن می پردازد و کد برنامه و فلوچارت های متعدد را در بر می گیرد و مشتمل بر نمایش اطلاعات تصویری و ویدئویی است می باشد . تاثیر این نرم افزار بر جامعه و فرهنگ بسیار زیاد است و باید سعی شود که در ساختن برنامه های کامپیوتری ارزان بودن و ساده به جواب رسیدن مد نظر باشد.

نقش تکاملی نرم افزار و بررسی آن :

امروزه نرم افزار نقش دوگانه ای دارد خود یک محصول است و. در عین حال وسیله ای برای ساختن و تحویل محصول به حساب می آید از نرم افزار اگر به عنوان وسیله انتقال دهنده استفاده شود بصورت پایه و اساس کامپیوتر است به منظور شناخت یک نرم افزار و نهایتاً دریافتن موضوع کارکرد آن خلاقیت انسان ، تخیل و طراحی و در نهایتاً شکل فیزیکی نرم افزار بسیار مهم است طبق استاندارد DIN هزینه های نرم افزاری متمرکز شده و این بدین معنی است که پروژه های نرم افزاری را نمی توان بصورت پروژه های نرم افزاری (که تولید مدیریتی ندارند) مدیریت نمود . شکل ۱ منحنی شکست سخت افزار را نسبت به زمان از لحاظ کهنه شدن نرم افزار نشان می دهد .



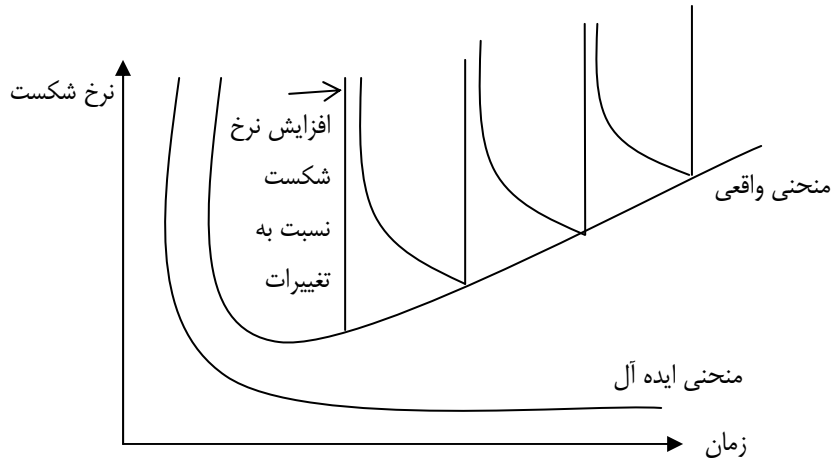
(شکل ۱)

به منحنی فوق منحنی وارن می گویند نشان می دهد که سخت افزار در اوایل عمرش چه مقدار موفقیت داشته که با گذشت زمان شروع به فرسوده شدن کرده و دوباره میزان شکست افزایش می یابد در نرم افزار عوامل محیطی سخت افزار تحت شعاع خود قرار داده و سختی کار و عدم موفقیت را باعث می شود شکل ۲ نشان می دهد که مشکلات شناسایی نشده در اوایل کار نرم افزاری باعث عدم موفقیت می شود و پس از برطرف شدن آنها اوج گرفته و نرم افزار فرسوده نمی شود .

مهم :

جنبه دیگر فرسودگی تفاوت بین سخت افزار و نرم افزار را موقعی می توان نشان داد که جزء سخت افزاری ما فرسوده شده و جایگزین می شود .

پس نگهداری از نرم افزار پیچیدگی بیشتری دارد به شکل ۲ نگاه کنید :



(شکل ۲)

با توجه به شکل در می یابیم که منحنی ایده آل و واقعی شکست در یک نرم افزار چگونه است و اجزا در یک نرم افزار طوری باید طراحی شود که قابلیت استفاده مجدد را داشته باشد طبق استاندارد IEEE کاربردهای نرم افزارهای Real Time ، نرم افزارهای تجاری و نرم افزار های مهندسی تقسیم می شود .

نرم افزارهای Real Time (بلادرنگ) :

این نرم افزار نظارت و تحلیل کنترل رویداد را به عهده دارد و به آن بلادرنگ می گویند مؤلفه ای که تبدیل این اطلاعات را به عهده دارد بگونه ای است که بتوان اجزاء خروجی را کنترل کرد و معمولاً بر حسب میلیونیم ثانیه می باشد . نرم افزارهای Real Time با سیستم عامل Real Time کار میکند و اکثراً بر روی عاملهای معمولی قابل اجرا نیستند سیستم عامل RSTX88 ، RSTX86 دو زیر پایه Real Time است .

نرم افزارهای تجاری :

بزرگترین حوزه برنامه های کاربردی نرم افزاری است که به صورت پایگاه چند لایه ای قابل دستیابی بوده و مثلاً به عنوان پردازش تراکنش های فروش یک فروشگاه قابل کار است .

نرم افزارهای مهندسی :

این نرم افزارها الگوریتم های پردازشگر عددی هستند که تحلیل فشار ترمز خودرو تا مقدار دمای گدازه آتش فشان را می توان محاسبه کرد .

استاندارد IEEE نرم افزارها را برحسب کاربرد خاص نیز دسته بندی می کنند که با عناوین :

۱) Personal Software : نرم افزار برای کامپیوترهای شخصی
۲) Web base Software : نرم افزارهایی که دارای پایه وب هستند .
۳) Haypertake Software : نرم افزارهایی که برحسب انتقال داده ها کار می کنند .
۴) Inteligent Software : نرم افزارهایی که با هوش مصنوعی کار می کنند .
۵) نرم افزارهای جایگزین

همگی اینها بر پایه (base) نرم افزاری بوده که بصورت تقریباً نامحدود بر روی هر کامپیوتر (با احتساب حافظه RAM) قابل دستیابی است .

نرم افزارهای جاسازی شده یا نرم افزارهایی که دارای هوش مصنوعی هستند :

فرآیندهای کوچک را به خوبی کنترل می کنند مثلاً صفحه کلید کنترل ماکرو و صفحه کنترل سوخت دیجیتال در خودرو .

نرم افزارهای کامپیوتر شخصی:

بازار کامپیوتر بسیار پررونق است نرم افزارهای مدیریتی مالی و تجاری انواع بازی و برنامه های تحت وب را شامل می شود .

نرم افزارهای پایه وب :

می دانید نرم افزارهایی که دستورات قابل اجرایی مثل : Java , Html , CGI و فایل های سمعی بصری را اجرا می کنند تحت نرم افزارهایی هستند که قابل دسترسی با مودم است یعنی می تواند آنها را از یک پایگاه دانلود کرد و با گرفتن کلید کاربردی Register و داشتن سند مالکیت آن نرم افزار به استفاده از آن پرداخت .

نرم افزارهای هوش مصنوعی یا Inteligent Software:

از یک سری الگوریتم غیر عددی بصورت شبکه های عصبی استفاده می کند که محاسبات پیچیده ای داشته و اشتباهات کوچک نیز به سختی اصلاح می شود شبکه عصبی رابطه مستقیم با کارکرد دیجیتال داشته و الگوریتم های کهنه Ras

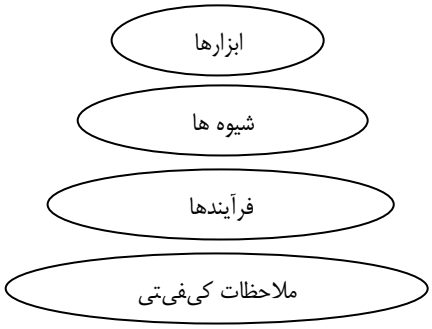
پس اینگونه می توان نتیجه گیری کرد که نرم افزار عنصر اصلی به سیستم های مبتنی به کامپیوتر است و تقریباً در ۵۰ سال گذشته نرم افزار مهمترین وسیله ارتباط اطلاعاتی ، تخصصی با یک صفت دیگر است هدف درس مهندسی نرم افزار نیز تامین چهارچوبی برای ساخت نرم افزارهای با کیفیت و کاربرد بیشتر است .

بررسی فن آور مهندسی نرم افزار به عنوان یک فن آوری لایه ای :

ایجاد و اسفاده از اصول ساده مهندسی در رسیدن به یک نرم افزار مقرون به صرفه و قابل اطمینان روی کامپیوترهای پرسرعت این تعریف اهمیت زیادی از نظر سرعت برای کامپیوتر قادر است لذا اگر کامپیوتری کم سرعت بوده و حافظه جانبی کمی داشته باشد قابلیت اجرای نرم افزار را ندارد و یا به کندی اجرا می کند . از نظر استاندارد IEEE به کارگیری یک روش سیستماتیک منظم و داشتن کیفیتی خوب و کمیتی قابل ارائه در شیوه های استفاده از یک نرم افزار و بررسی رهیافت های اجرای آن شده است . اینگونه می توان استنباط کرد که استفاده از یک نرم افزار کارایی بهینه و بهترین حالت آن موقعی اتفاق می افتد که کاربر راه کار کردن با آن را بداند و ابزارهای آن را بشناسد . گفتیم که اساس مهندسی نرم افزار یک فرآیند لایه ای است پس همانند چسبی لایه های فن آوری را با هم نگه می دارد و تحول ایجاد مجموعه ای کلیدی و کارآمد را به وجود می آورد.

استفاده از اسناد :

داده های بیرون آمده از دیگر نرم افزارها کارکرد نرم افزار را بهینه می کند داشتن یک ابزار مهندسی مناسب در یک نرم افزار و همچنین پشتیبانی اتوماتیک یا نیمه اتوماتیک برای انجام فعالیت ها انسجام خاصی به ابزارها داده که طبق نظر استاندارد IEEE به مهندسی نرم افزار مهندسی کمک کامپیوتر نیز می گویند . استفاده از ابزارها شیوه ها و فرآیندها در بستر ملاحظات کیفیتی باعث به هم نگهداشتن لایه های مهندسی نرم افزار می شود به شکل ۳ نگاه کنید .



(شکل ۳) لایه های مهندسی نرم افزار

تعریفات اساسی:

در مبحث مهندسی نرم افزار یک سری تعاریف اساسی و کلیدی وجود دارد که به قرار زیر است :

- ✓ **اصلاح پذیری :** در مورد بهترین کارها نیز می توان گفت که کار بهتر نیز وجود دارد و باید برای یک نرم افزار ورژن های مختلفی از بسته اصلی به وجود آید تا هر ساله بسته به نیاز کاربر به روز شده و قابلیت آن نیز اضافه شود و نواقص آن کمتر گردد .

- ✓ **تطابق :** به مرور زمان محیط اصلی یک نرم افزار که برای یک cpu وسیستم عامل خاصی طراحی شده است . برای سیستم قدیمی می شود یک نرم افزار باید قابلیت انجام کارهای تطابقی که منجر به تغییراتی از نرم افزار می شود را با حفظ محیط خارجی اش داشته باشد . به عنوان مثال نرم افزاری موفق است که در حین اجرا به روی win98 قابلیت بارگذاری در محیط Vista , XP را نیز داشته باشد .

- ✓ **بهبود وضعیت :** با استفاده از نرم افزار مشتری باید بتواند کاربردی موفق به حساب آید یعنی از نرم افزار به صورتی خوب و بی عیب استفاده کنند هم چنان که مهندس نرم افزار این کار را انجام می دهد با گذشت مقدار کمی از زمان از عرضه نرم افزار به بازار استفاده کنندگان به طراح خبر کارکرد مطلوب نرم افزار را می دهند .

- ✓ **پیشگیری :** نرم افزارهای کامپیوتری با کوچکترین تغییراتی در Setting آنها کارکردشان عوض می شود که اغلب کاربران نسبت به آن بی اطلاع اند باید بتوان کاری کرد بر روی یک نرم افزار موفق شرایط اصلی کارکرد نرم افزار بصورتی از قبل تعیین شده تغییر کند و همچنین به سیستم Setting اولیه به راحتی برگردد مطالب یاد شده این را می گوید که استفاده کننده از نرم افزار نباید بتواند کلیات سیستم نرم افزار را تغییر دهد بلکه بصورت جزیی تغییرات را اعمال کند تا پیشگیری از خطرات ثانویه نماید .

کامل شدن فرآیند :

مؤسسه مهندسی SBI آمریکا مدل جامعه ای ارائه کرده که روی مجموعه ای از توانایی های مهندسی نرم افزار که سطوح بلوغ مختلفی برای آن در نظر گرفته شده است را قابل دسترس می کند . این انیستیتو یک طرح ۵ درجه ای امتیازی در میزان تطابیق توانایی CMM نرم افزارها مشخص می کند که فعالیت های لازم در سطوح مختلف یک فرآیند را تعریف می کند . ۵ سطوح به قرار زیر است :

۱- **سطح اول (اولیه) :** یک فرآیند نرم افزاری بصورت موقتی و گاهی اوقات بصورت قانونی برای هم توصیف می شوند که این فرآیند به موفقیتی می رسد که به تلاش های فردی اهمیت داده شده و به آن بستگی دارد .

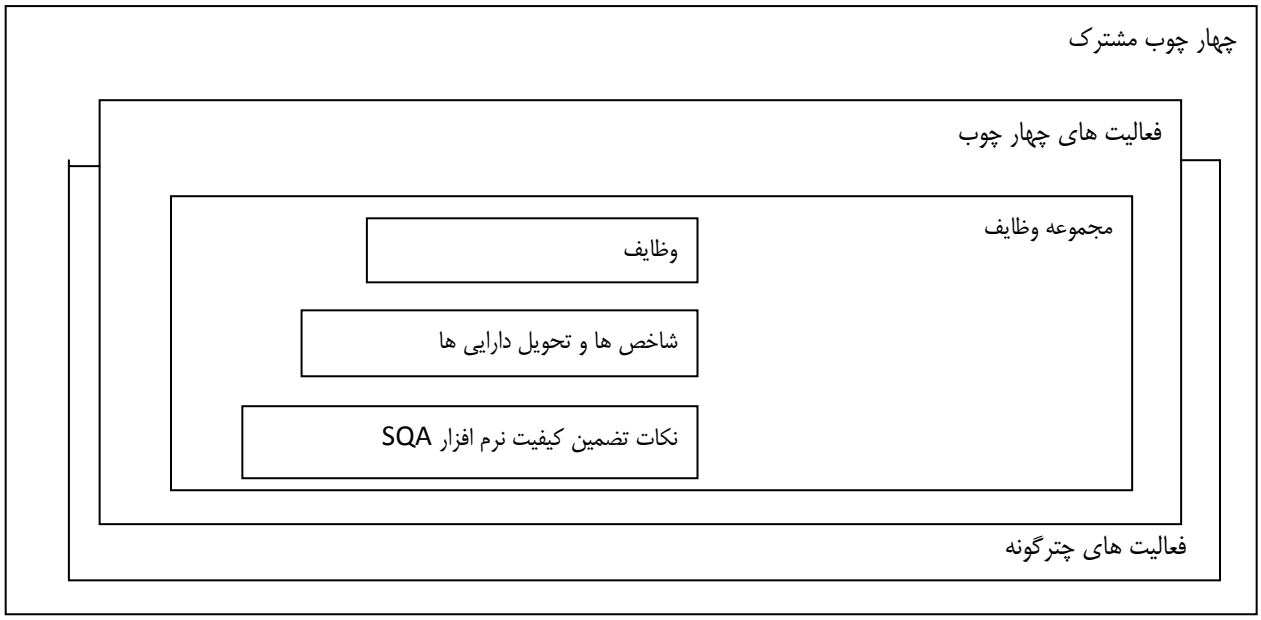
۲- **سطح قابل تکرار :** فرآیندهایی که بصورت اولیه یک مدیر پروژه برای زمانبندی کارایی و حدس زدن هزینه ها استفاده می کند . باعث ایجاد موقفیت هایی نسبت به پروژه های موازی می شود .

۳- **مقدار تعریف شده :** برای مدیریت و فعالیتهایی که در طول یک فرآیند که دارای استاندارد خاصی می باشد صورت می گیرد و باید هم یک نسخه تایید شده از آن نرم افزار وجود داشته باشد و این نسخه بصورت پشتیبان تمامی فرامین نرم افزاری را ساپورت نماید .

۴- **بهینه سازی فرآیند :** بهبود در بهینه سازی کارکرد یک نرم افزار به Feed back آن نرم افزار از روی فکرها و ایده هایی که با استفاده از فن آوری های نوین و نوآورانه مقدور باشد استفاده می شود . پس باید هم از وقت و هم هزینه و فن آوری ها طوری استفاده شود تا پس از مدت زمان معینی نرم افزار به سطح قابل اجرایی بصورت بهینه برسد .

۵- **از یک نرم افزار و پردازش آن کارهایی که در کیفیت محصول مهم هستند** باید شناسایی شده و چه از لحاظ کمی و کیفیتی با استفاده از روش هایی کاربردی یک کنترل دقیق صورت بگیرد تا بتوان به بهترین حالت به مدیریتی مناسب دست پیدا کرد .

نکته : پس از نگاه کردن به شکل ۴ در می یابیم که هر سطحی که بالای سطح ۲ قرار دارد یک اشتراک و زیر مجموعه از سطح قبلی است پس این گونه استنباط می شود که یک تولید کننده نرم افزار در سطح ۳ باید به سطح ۲ رسیده باشد تا بتواند به فعالیتهای مناسب در سطح خود دست یابد .



(شکل ۴) فرآیند نرم افزار

نکته :

با توجه به شکل و کتاب های آسیایی می توان سطوح ۴و ۵ را در بهینه سازی و سیستم های مدیریت شده طوری ادغام کرد که هر یک اثری از دیگری داشته باشد .

۵ سطح تعیین شده توسط ESI باعث واکنش هایی شده است که در حوزه فرآیند به KPA معروف است این KPA ها می توانند فعالیت مهندسی نرم افزار توصیف کنند و نیازمندیهای مدیریتی را که برای انجام کار لازم است توصیف می نمایند که شامل اهداف ، تعهدات ، توانایی ها و فعالیت های KPA می باشد .

کنترلهای روش های پیاده سازی در نرم افزار و شیوه های KPA براساس عملکرد آنها تعریف می شود که در زیر به ۵ سطح دسته بندی می گردد .

سطح اول : سطح دسترسی می باشد .

سطح دوم : شامل فرآیند تکمیلی است که خود دارای زیر مجموعه های زیر است :

- ✓ مدیریت پیگیری نرم افزار
- ✓ تضمین کیفیت نرم افزار
- ✓ مدیریت پیمانکاران نرم افزار
- ✓ پیکر بندی و نظارت پروژه ها
- ✓ مدیریت نیازمندیها

سطح سوم : که به سطح تکمیلی طبق استاندارد IEEE معروف است دارای سطوح زیر است :

- ✓ بازبینی های یکسان
- ✓ همکاری های درون گروهی
- ✓ مهندسی محصول
- ✓ مدیریت یکپارچه سازی
- ✓ برنامه های آموزشی
- ✓ استفاده از تعریف فرآیند سازمانی در برنامه ها و نقطه تمرکز سازمانی

توجه :

در اکثر CMM ها به ۵ سطح مختلف و لایه های مختلف تقسیم بندی می شوند که در سایت www.Sepo.Nose.com قابل دیدن است.

سطح چهارم : دارای لایه های زیر است که مدیریت کیفیت نرم افزار و مدیریت فرآیند کمی می باشد .

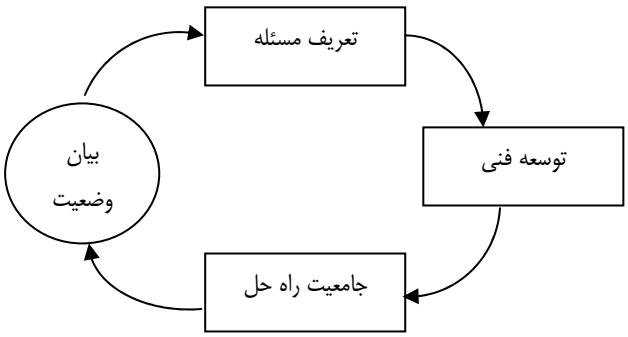
سطح پنجم : به فرآیند تکمیلی معروف است و دارای دو زیر سطح:

- ✓ مدیریت تغییر فرآیند
- ✓ پیشگیری از اشکال است.

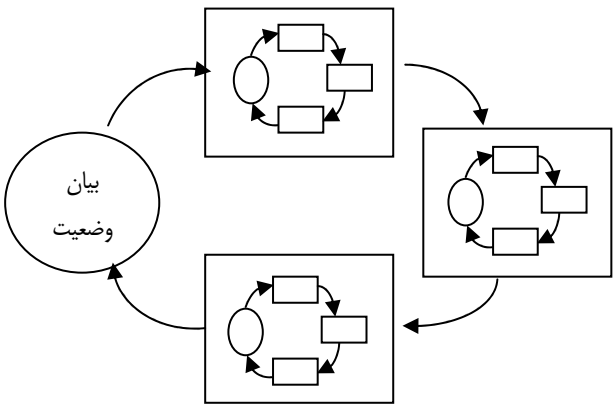
که هریک از KPA ها توسط یک سری روشهای مهم تعریف شده و به امر دستیابی به هدف کمک می کند .

مدل فرآیند یا پارادایم :

از مدلهای مهندسی نرم افزار نشان می دهد که ماهیت پروژه و کاربرد آن در روشهای مورد استفاده و کنترل قابل ارائه کدام است این مدل نشان می دهد که تمامی کارهایی که توسط یک نرم افزار به عنوان یک حلقه تشکیل می تواند صورت بپذیرد چگونه دیده می شود . مسئله را طبق نظر استاندارد (SEI) می توان با دو روش بررسی چرخش کامل مسئله و بررسی مراحل میانی چرخش مسئله را حل کرد و کل مراحل و فازهای مهندسی نرم افزرا را در آن ترسیم نمود به مدل پارادایم زیر نگاه کنید :



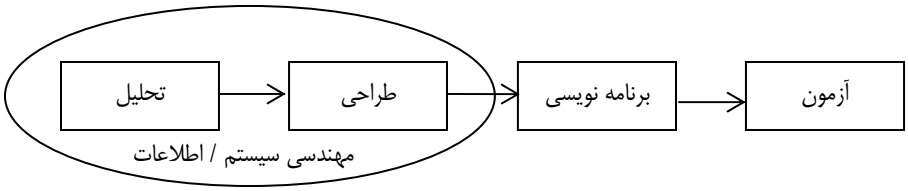
(شکل ۵ الف) اصلی RAC95



(شکل ۵ ب) میانی RA95

مدل ترتیبی خطی :

این مدل که گاهی به چرخه حیات کلاسیک معروف است و در کتابهای آسیایی به آن مدل آبشاری می گویند . این مدل بیانگر یک روش نظام مند و زنجیروار است که در نهایت به تولید نرم افزار ختم شده و در یک سطح شروع شده با تحلیل طراحی و کد گذاری به همراه آزمون و پشتیبانی پیشروی می کند شکل ۶ یک مدل ترتیبی کلاسیک را با استفاده از استاندارد IEEE نشان می دهد به شکل نگاه کنید:



(شکل ۶) مدل ترتیبی خطی

در آن تحلیل ، طراحی ، برنامه نویسی و آزمون را می بینید درک ماهیت برنامه تولیدی در مهندسی نرم افزار باید با دامنه اطلاعات مورد نظر ارتباط داشته و عملکرد را بالا ببرد.

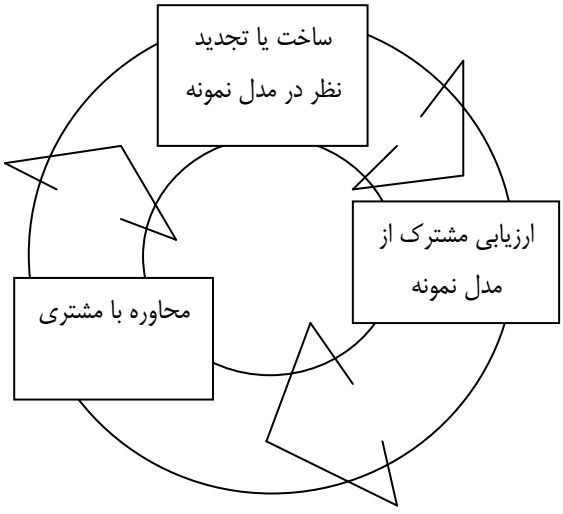
تحلیل : همانطور که گفتیم در جمع آوری نیازمندیها در مهندسی نرم افزار دامنه تحلیل مهندس در برخورد با اطلاعات و مسائل پیش رو به ضریب هوشی و درک ماهیت بالایی احتیاج دارد.

طراحی : طراحی نرم افزار در حقیقت یک فرآیند Step by Step که روی ۴ صفت متمایز یک برنامه Zoom می کند یعنی ساختار داده ها ، معماری نرم افزار نمایش ارتباطات و جزئیات رویه ها یا الگوریتم ها که باید از قبل آن را کد گذاری کرده بشناسیم.

ایجاد کد یا کدگذاری : طراح باید به صورتی کار کند که مراحل اجرای کار به صورت دقیق صورت گیرد و کدها کاملاً بی اشکال و رُند باشند . حال وقتی کد ایجاد شد آزمون برنامه شروع می شود که کار آزمون روی فواصل زمانی منطقی در نرم افزار متمرکز می شود . مدل نمونه سازی یا پارادایم(Prototyping) در بعد توضیح داده می شود .و اگر خطاها زیاد بوده یا با نتایج لازم در خروجی ها هم خوانی نداشته باشند طراح باید آن را عوض کند.

مدل نمونه سازی یا پارادایم :

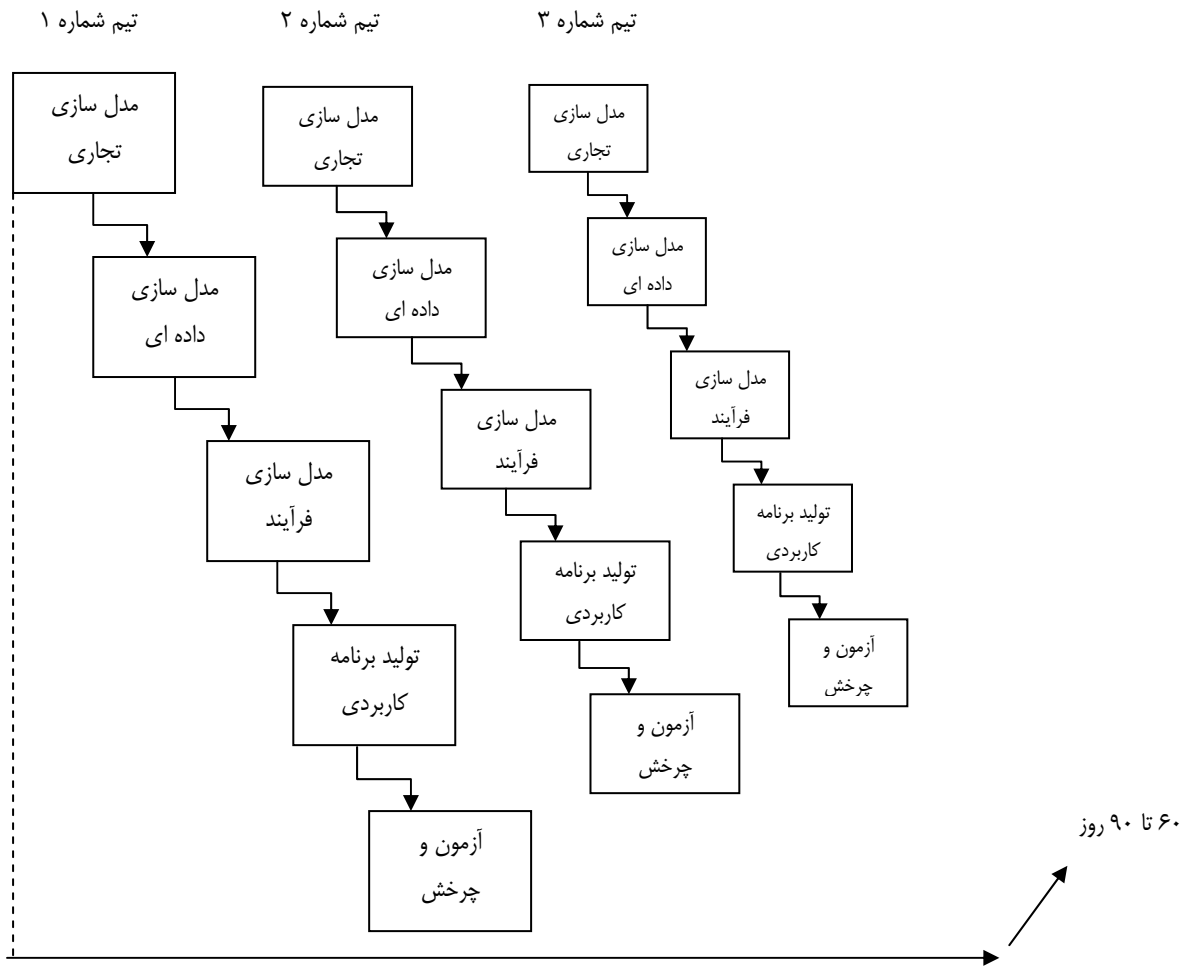
اغلب مشتریها برای خرید نرم افزار ابتدا نیازهای خود را بررسی می کنند و سپس به الگوریتم ساخت برنامه رجوع می کنند تا فلوچارت و قابلیت تطابق نرم افزار را با نیازهای خود پیدا کنند . پارادایم نمونه سازی همانطور که در شکل ۷ می بینید با جمع آوری نیازمندیها شروع می شود و با ارزیابی مشتری از مدل نمونه و استفاده از نرم افزار در حال اجرا ، برای ارزیابی و رسیدن به کارایی مورد نظر صورت می گیرد . اگر مهندسین طراح نیز در آنجا حضور داشته باشند .با یک محاوره ساده با مشتری خواسته های او را در یافته او را انتخاب نرم افزار یاری می کند به پارادایم ساخت نمونه طبق نظر استاندارد IEEE نگاه کنید .



(شکل ۷) پارادایم ساخت نمونه

مدل ساخت سریع برنامه ها یا RAD :

این مدل یک فرآیند افزایشی تولید نرم افزار است که با استفاده از تولید سریع و به نتیجه رساندن پروژه کار می کند این مدل کمک می کند که گروه طراح مدل پیشنهاد شده خود را ظرف مدت کوتاهی ۶۰ ، ۹۰ ، ۱۲۰ روزه به انجام رسانده و آماده اولین آزمایشات شود به مدل توسعه سریع RAD نگاه کنید .



(شکل شماره ۸)

مدل سازی تجاری :

این مدل سازیدر حقیقت نشان دهنده وضعیت تجاری نرم افزار در رقابت با سایر نرم افزارها ست و اینکه اطلاعات نرم افزاری چگونه کار می کنند و چه اطلاعاتی تولید می شود و این اطلاعات به کجا می رود و در نهایت نتیجه آن چه می شود را مد نظر دارد . این مدل سازی با استفاده از روشهای پیش بینی فرآیند بازار به سازندگان کمک می کند که عملیات تولید و عرضه را با پیش بینی های انجام شده ، انجام داده و شرکت را بیشترین مقدار سود برساند .

مدل سازی داده ای :

این مدل سازی در مهندسی نرم افزار کمک می کند که سازنده در هر مرحله از انجام ساخت داده های قبلی را نسبت به داده های مورد استفاده (داده های هدف) بررسی کرده نسبت استفاده آنها را در نسخه اولیه نرم افزار با استفاده از روش سعی و خطا و با بهره گیری از سیستمهای R&D سازمان پیش برده تا به بهترین هدف برسند .

مدل سازی فرآیند :

ما سعی می کنیم در ساخت برنامه ها از یک مدل مرجع استفاده کنیم و پیشرفت را نسبت به آن مدل بسنجیم در فرآیند مدل سازی استفاده از یک مدل جهت پیشرفت فرایند تولید بسیار کارآمد است چون مهندسین را وادار می کند تا اشکالات نرم افزار را نسبت به مدل مرجع برطرف سازند .

تولید برنامه های کاربردی:

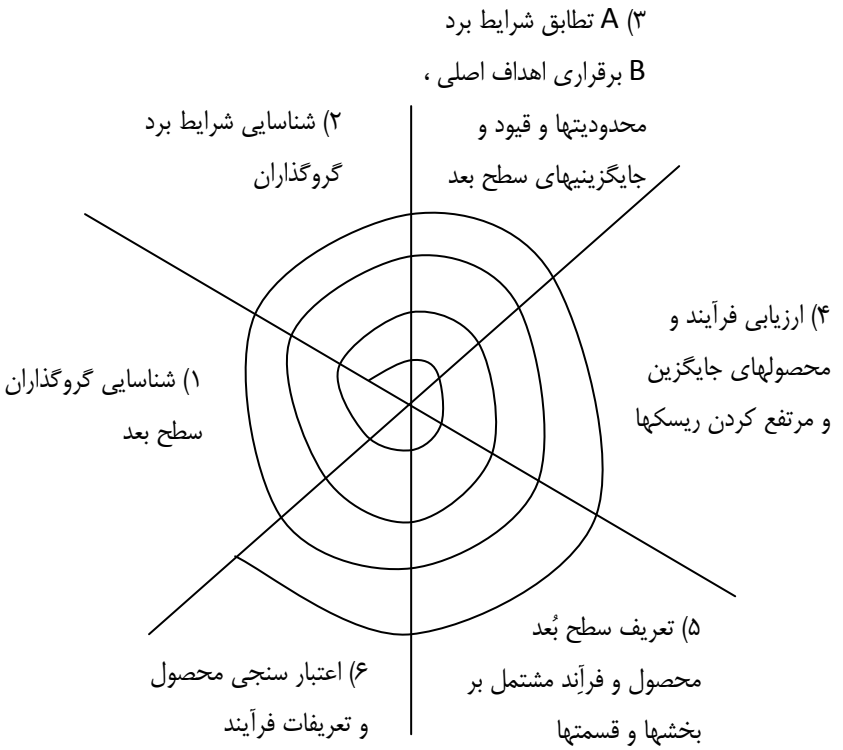
مردم RAD در حقیقت سعی دارد تا با استفاده از نرم افزارهای نسل چهارم برنامه هایی تولید کنند که با انواع سیستمهای عامل سازگاری داشته و بتواند در روی اکثر کامپیوترها (۳۲ بیتی و ۶۴ بیتی) به راحتی نصب شده و کار کند .

آزمون و چرخش :

در مدل RAD سعی شده است تا اکثر گروههای کاری در نقش نرم افزار در دوره های کاری مشخص شده پس از انجام آزمون خطا گروهها به صورت چرخشی با هم عوض شوند تا طبق نظر RAD کارایی کل پرسنل بالا رفته و همیشه در بیشترین مقدار قرار داشته باشند

بررسی مدل حلزونی WinWin (برنده ، برنده)

این مدل در مهندسین نرم افزار با تکیه بر نظر مشتری (مشتری مداری) کار می کند این گونه که سازنده با دادن نسخه beta نرم افزار به بازار سعی می کند نظرات مصرف کنندگان را نسبت به نرم افزار مطلع شده از بهترین آنها برای تولید نسخه اصلی استفاده کنید . نسخه Beta بین ۹ تا ۱۵ ماه قبل از تولید نسخه اصلی طبق نظر استاندارد به بازار عرضه می شود این مدل به دلیل این برنده برنده نام گرفته است که مشتری با به دست آوردن بهترین محصول بهترین کارایی را از آن می برد ، به مدل حلزونی winwin نگاه کنید .



(شکل ۹)

در اصل نقاط نمایشگر ما ۳ دیدگاه مختلف را (از ۳ بُعد) طی می کند که به آن LOLCO (چرخه حیات) می گویند . این مجموعه از اهداف برای هر فعالیت عمده ای بصورت LCO درآمده و مجموعه اهداف مرتبط را از نیازهای سطح بالاتر (در بُعد دیگر) حاصل می کند. معماری چرخه حیات یا LCA تیم پروژه نرم افزاری را وادار می کند که قابلیتهای بالاتری داشته باشد سومین تکیه گاه IOC یا قابلیت عملیات اولیه است که مجموعه اهداف مربوط به آماده سازی نرم افزار برای نصب را انجام می دهد این مدل از سال ۹۴ به بعد کمتر استفاده شده است . مدل winwin ۳ معیار فرآیند پردازش دارد که همانطور که ذکر شد تصمیم گیری های لازم را برای انجام بهتر پروژه و جایگزین کردن ریسکها با مقادیر کمتر انجام می دهد . در این دیاگرام برنده بودن سطح دلپذیر است و شناسایی سطوح با شناسایی گروگذاران در نرم افزار اولیه (نفرات شرکت کننده در نظر سنجی اولیه) با استفاده از نظرات گروگذاران تطابقی جهت برقراری شرایط بردن با استفاده از رسیدن به سطح بالاتری از موفقیت قابل دستیابی است . داشتن تعریف دقیقی از بُعد و کارایی سیستم حلزونی winwin این امکان را برای سازندگان محصول طوری فراهم می کند که مشتریان اغنا کند محصول آنها دارای سطح بالایی از فرآیندهای قابل اطمینان است (اعتبار و کارایی بالایی دارد) تا مورد استفاده قرار گیرد.

الگوهای سازمانی مهندسی نرم افزار:

کونستانتین سمیر ۴ الگوی سازمانی را برای تیم های مهندسی نرم افزار به استاندارد ANSI پیشنهاد کرد که در آگوست ۹۲ مورد قبول واقع شد . این الگوها شامل :

- ✓ الگوی بسته
- ✓ الگوی تصادفی
- ✓ الگوی باز
- ✓ الگوی همنام

الگوی بسته :

این گروه در حقیقت الگویی وابسته به درون گروه است و نسبت به کارایی و توانمندی های گروه دیدی درون گرا دارد . این الگو در حقیقت دانسته های گروه را به base اصلی کار برای انجام فعالیتهای ساخت نرم افزار قرار می دهد و نسبت به دید بیرونی و استفاده از نظرات از بیرون گروه کلاً نظری بسته دارد .

الگوی تصادفی :

این الگو متوسط از توانمندی های گروه در ساخت نرم افزار با استفاده از روش تصادفی است که روش تصادفی در حقیقت در زیر مبحث نظم و انضباط در کار را شیوه های جدید ساخت مهندسی نرم افزار **با استفاده از عامل خلاقیت و ابداعات و نوآوری های سیستماتیک تکنولوژی یک الگوی تمام بسته را که دارای سلسله مراتب رأسی (هرمی) کار می کند .**

الگوی باز :

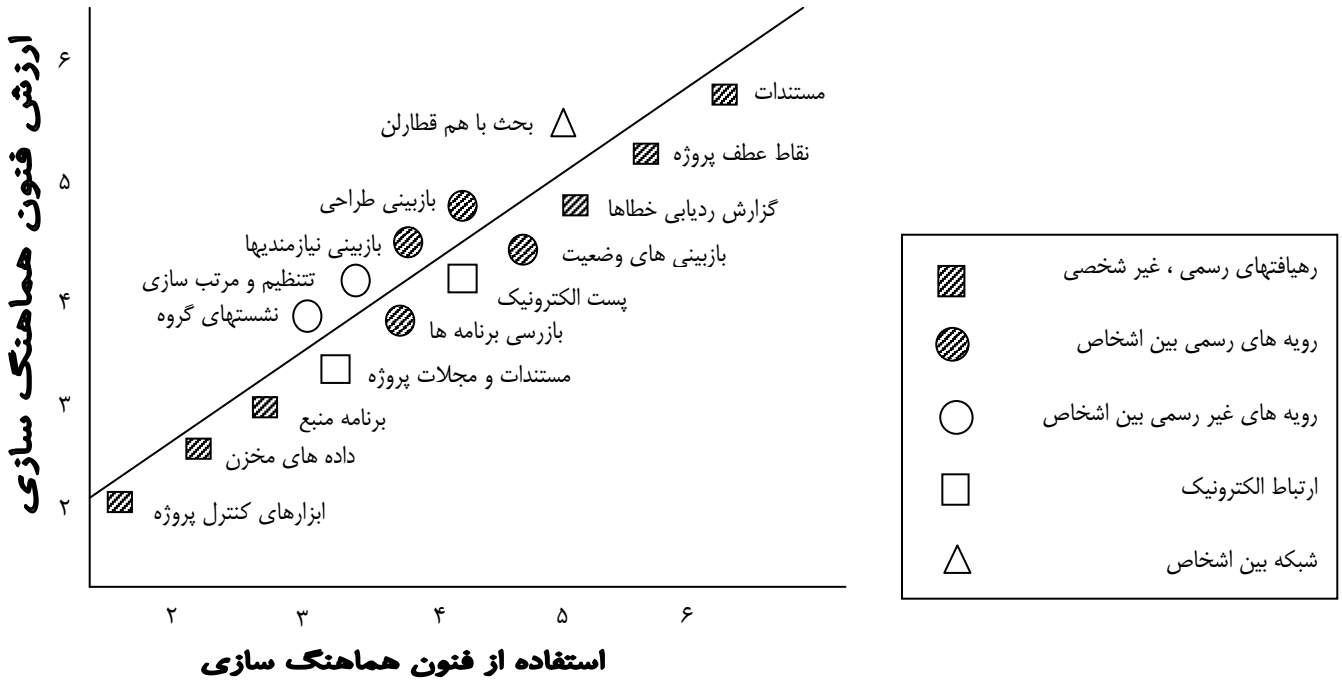
این الگو بر پایه نظرات پیشنهادی برای نسخه های Beta در نرم افزارهای نمونه است و این الگو با استفاده از کار دسته جمعی و نمونه های دیگر نرم افزار سعی به رسیدن بهترین نسخه و استفاده از نظرات باز در بهتر کار کردن نرم افزار خواهد بود برای حل مشکلات نرم افزاری در این روش head گروه با استفاده از انواع نظر سنجی در کار با نسخه های متشابه سعی در بهتر کردن کاربرد نرم افزار دارد.

الگوهای همنام :

این الگو بر پایه مهار کردن هرج و مرج تیمی بنا شده است در این الگو که نمونه طبیعی کار کردن موازی اعضای تیم است با استفاده از **اعتماد و مهارت در تیم گروه به کارهای موازی پرداخته تا در نهایت به هدف مشترک برسند .**توضیح اینکه در الگوهای همنام ممکن است در روی یک مبحث ۲ یا چند گروه کار کنند که بهترین برآیند تیمی مورد استفاده قرار گرفته (بهترین نتیجه) و تعداد کارکنان در این روش در تیم ها زیاد بوده و بیشتر برای شرکتهای بزرگ مناسب می باشد .

هماهنگی مسائل و ارتباطات :

مجموعه ای از قوانین هماهنگ شده را برای انجام پروژه های نرم افزاری برگزید که در سال ۹۰ استاندارد ANSI آن را به عنوان قوانین و فنون هماهنگی در پروژه های نرم افزاری برگزید ساختن نرم افزار دارای پیچیدگی و مشکلات خاصی است که اعضای تیم با استفاده از اشتراک و ارتباط محدودیتها را از راه برداشته و بهترین روش را برای ساخت یک نرم افزار انجام می دهند هماهنگی بین گروه ها برای پیشروی و رسیدن به هدف بسیار مهم است و این کار توسط افراد مسئول در رسیدن به بهترین هدف انجام می شود الگویی که در زیر می بینید یک الگوی کاربردی تحت نظر استاندارد ANSI که با استفاده از نظرات مشترک و کاربردهای علمی و عملی از دستاوردهای جدید تکنولوژی در رسیدن به بهترین هدف یاری می کند . به الگو نگاه کنید :



(شکل ۱۰) ارزش کاربرد فنون ارتباط و هماهنگ سازی

اقدامات بحرانی :

شورای ۷ نفره استاندارد ANSI لیستی از کارهای حساس و مهم نرم افزاری را که در ارتباط با مدیریت و عملکرد بهینه کارهایی را بصورت ثابت و پیوسته جهت بالا بردن کارائی و ضریب بهینه سازی نرم افزار در صنایع مورد استفاده قرار می گیرد . این شورا کارهای زیر را که ۵ عمل مهم در ساخت نرم افزار نام دارد اینگونه نام گذاری کرده است :

۱. مدیریت رسمی ریسک
۲. مدیریت پروژه مبتنی بر متریک ها

۳. هزینه عملی پروژه
۴. رهگیری مقادیر به دست آمده
۵. پیگیری معایب در مقابل اهداف
۶. مدیریت برنامه ها

۱. مدیریت رسمی ریسک :

این مدیریت ۱۰ خطر عمده برای هر پروژه باز می کند که رئیس پروژه باید آنها را بشناسد در مورد هر یک از احتمالات و احتمال وجود تبدیل آنها به خطر توسط رئیس پروژه و تاثیراتی که این خطرات بر روی نسخه Beta نرم افزار و همچنین اصلی آن می گذارند چیست .

۲. مدیریت پروژه مبتنی بر متریک ها :

این سؤال که آیا شما برنامه هایی دارید که علایم اولیه بروز مشکل را نشان دهد و به موقع نشان دهد و اگر مشکل پیش آید چگونه باید آنرا رفع کرد .

۳. هزینه عملی پروژه :

اندازه تخمین زده شده هزینه ها برای کاربرد نرم افزار چه مقدار است و این اندازه چگونه تعیین می شود را رئیس گروه باید پاسخ دهد.

۴. رهگیری مقادیر به دست آمده :

آیا شما اندازه به دست آمده را در ساخت نرم افزار به صورت روزانه چک می کنید و آیا این استانداردها باعث ایجاد تحول در base نرم افزار می شود یا خیر ؟ که رئیس گروه باید به آن پاسخ دهد .

۵. پیگیری معایب در مقابل اهداف :

معایب مشخص شده در حین هر بازرسی باید دنبال شود تا تعداد آنها در روزهای آخر پروژه به صفر نزدیک شود .

۶. مدیریت برنامه ها :

عملکرد هر یک از کارکنان در ماه گذشته در توسعه و بازیابی ساخت یک نرم افزار بهینه تاثیر گذار بوده و باید مدیران هر قسمت عملکرد کارکنان را به رئیس پروژه اطلاع داده و در صورت لزوم برای بهترین ها پاداش در نظر بگیرد.

اندازه گیری منابع محصولات در فرآیندها :

در خصوص این مسئله در مهندسی نرم افزار باید اندازه گیری متریک پروژه و جوانب کارایی پروژه سنجیده شود استاندارد IEEE چهار دلیل برای اندازه گیری منابع و محصولات آن ها به قرار زیر اعلام می کند :

- ✓ مشخص کردن
- ✓ ارزیابی کردن
- ✓ پیش بینی کردن
- ✓ پیشرفت کردن

مشخص کردن :

ما برای رسیدن به فهم و درک در مورد فرآیندها باید اندازه ها را مشخص کنیم و باید بتوانیم با داشتن خطی مقایسه ای پیشرفت را مشخص کرده و هم چنین اندازه خطاها را برای بهترین کارکرد نرم افزار تعیین کنیم . ریچارد فلوراک با مشخص کردن یک خط مبنا در مقایسه و ارزیابی فرآیندهای نرم افزاری بهترین روش را برای رسیدن به کمترین مقدار خطا و مشخص کردن آن در پایه نرم افزاری انجام داد.

ارزیابی کردن :

در ساخت نرم افزارهای مهندسی پس از انجام کارهای اولیه گروه نرم افزاری باید بتواند پیشرفت کار خود را ارزیابی کند این ارزیابی توسط مقایسه با نرم افزارهای دیگر و همچنین استفاده از نسخه Beta نرم افزار برای رفع نقص احتمالی درآن است ضمناً ارزیابی به گروه کمک می کند که نتیجه کار خود را بهتر سنجیده و برای بهتر شدن آن تلاش کنند .

پیش بینی کردن :

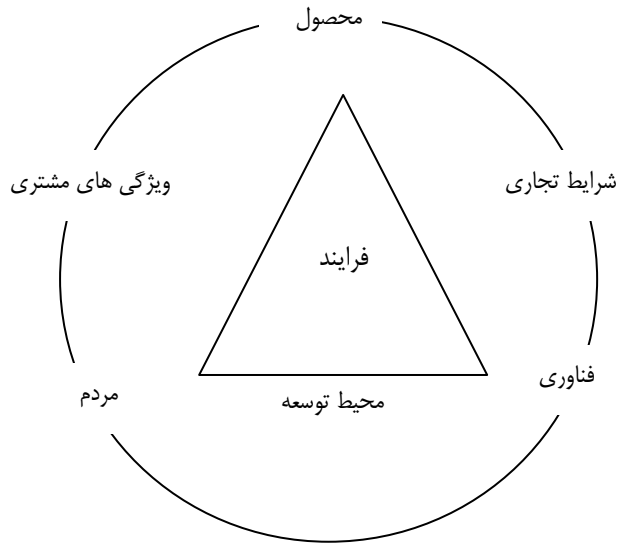
ما پیش بینی می کنیم تا بتوانیم طرح ریزی کرده و کار را به نحو احسن انجام دهیم این پیش بینی مستلزم ارتباطات میان گروهی بوده و برای رسیدن به هدفی متمرکز با استفاده از تخمین هزینه ها ، تخمین مشکلات پیش رو و تخمین زمان رسیدن به هدف می توان گروه نرم افزاری را با استفاده از این پیش بینی ها در رسیدن به یک هدف متمرکز کمک کند . گروههای نرم افزاری در ساخت نسخه های اولیه نرم افزارها غالباً با مشکلاتی روبرو می شوند که با استفاده از تدابیری که از پیش تعیین کرده اند سعی در رفع نقایص دارند .

پیشرفت :

برای اندازه گیری پیشرفت باید اطلاعات کمی و کیفی با حضور شاخه های کاری موجود باشد تا پیشرفت را نسبت به نقطه اولیه برای رسیدن به نقطه هدف در مدت زمان کار پروژه حاصل می گردد در گروههای نرم افزاری داشتن ریشه های اصلی در پیشرفت و شناساندن نقیصه های نرم افزار به گروه کمکم بزرگی کرده و استاندارد ANSI استفادهن از نرم افزارهای پیشرفت پروژه را که از نظر زمان و هزینه کارایی را بالا می برند مجاز دانسته است دو نرم افزار MS Project و Premavera در به سرانجام رساندن پروژه ها با استفاده از ارزشیابی نقطه به نقطه کارایی بالایی داشته و در انجام عملی پروژه به کارفرما و رئیس پروژه کمک کرده و نقطه بهینه پیشرفت را به صورت State By State عرضه می دارد .

مؤثر بودن فرآیند یک نرم افزار:

چگونه می توانیم مؤثر بودن یک فرآیند نرم افزار را اندازه گیری کنیم در پاسخ به این سؤال باید گفت که ۳ عامل با تأثیر زیاد بر کیفیت نرم افزار عملکرد سازمانی و مهارت و انگیزه بر مؤثر بودن یک نرم افزار تأثیر دارد استاندارد ANSI با استفاده از روشهای مهندسی نرم افزار در سراسر مراحل پردازش نظارت داشته و یک مثلث فرآیند را در سراسر مراحل پردازش در میان یک دایره شرایط محیطی که شامل محیط توسعه شرایط تجاری و ویژگی های مشتری می باشد (ابزار ، قوانین تجاری ، آسان کردن ، ارتباطات مشتری با نرم افزار می شود .) به پارادایم تعیین کیفیت نرم افزار طبق استاندارد ANSI نگاه کنید :



(شکل ۱۱)

می بینید که کارایی یک فرآیند به روش غیر مستقیم اندازه گیری شده به نحوی که با استفاده از نتایج و متریک های بدست آمده از قبیل اندازه خطاها و بررسی معایب که به گروه منتقل شده است بهره وری در نرم افزار را بالا برده و از طریق اندازه گیری های ویژگی مهندسی نرم افزار یک فرآند کامل که از ۳ بخش خریدار ، محصول و فرآیند استنتاج شده است را با استفاده از حداقل زمان و بیشترین مقدار تلاش فعالیتهای مهندسی نرم افزار را اندازه گیری کنیم .

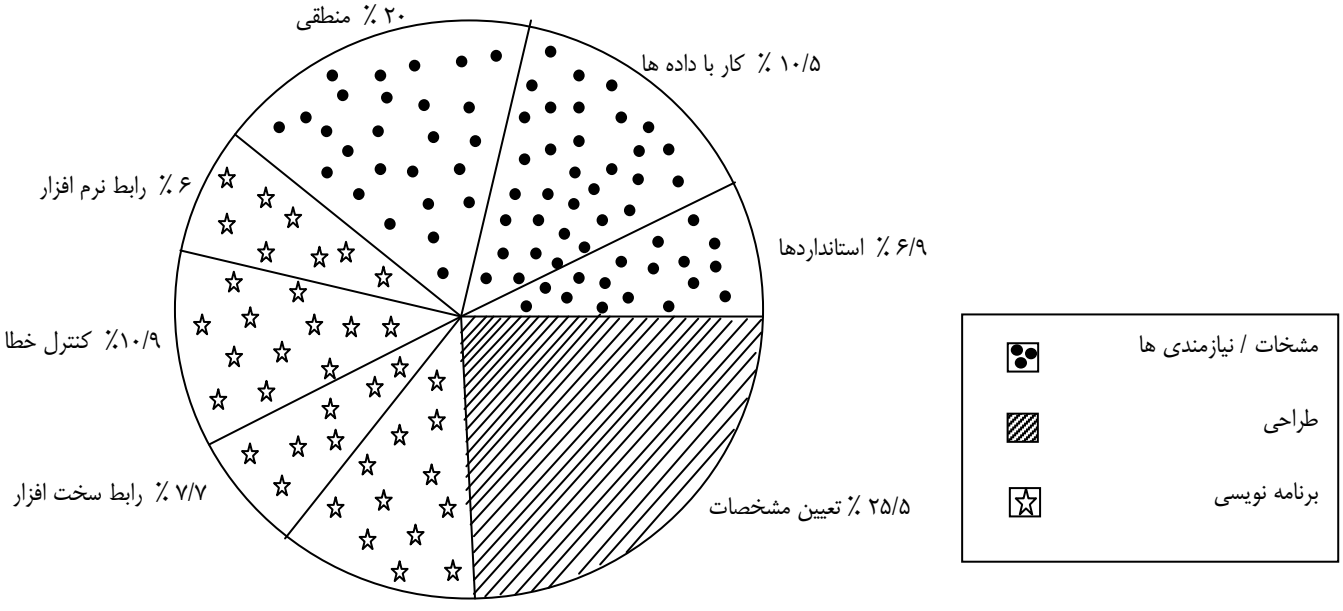
رابرت جی گریدی استدلال کرده است که کاربردهای عمومی و خصوصی بر انواع داده پردازی های مهندسی نرم افزار با استفاده از متریک هایی مانند نرخ معایب و خطاهای پیشرفتی قابل اندازه گیری است و با استفاده از نظرات گریدی ، جان هانفری یک رهیافت توسعه یافته در دانشگاه لویزانا از فرآیند نرم افزار شخصی (Personal Software Process) PSP که مجموعه ای از ساخت یافته های توصیف شده پردازشی و اندازه گیری های مؤثر در فرآیند پیشرفت مهندسی نرم افزار می باشد . فرآیند PSP با همه فرآیند های اندازه گیری متفاوت است این فرآیند برای هر مهندس نرم افزار **در اندازه گیری کار خودش مؤثر می باشد و با این کار به بهتر شدن روش ساخت نرم افزار کمک می کند .** (با استفاده از متریک های شخصی PSP راهبری کل ساخت نرم افزار دچار تغییرات تکنیکی و فنی کمتر شده و با به روز نگاه داشتن مهندس نرم افزار به پیشبرد کارایی گروه و دانش ساخت نرم افزار کمک می کند .)

توجه :

پیشبرد فرآیند نرم افزاری باید در سطح فردی آغاز شود و داده های پردازشی خصوصی هنگامی که در حرکت به سوی بهبود وضعیت سطح دانش مهندس نرم افزار است کمک کرده تا عملیات کل گروه بهبود یابد .

دلایل عیوب و منشأ آنها در پروژه های نرم افزاری:

گریدی در سال ۹۷ استدلال کرد که متریک های فرآیند نرم افزار و دلایل عیوب آنها دلایلی هستند که توسط سازمان در کنترل داده ها از قبیل کنترل خطا ، رابط های سخت افزاری ، رابط های نرم افزاری ، استنباطات منطقی و همچنین سطح استفاده از استانداردها در کار با استفاده از یک عملیات منطقی در پروژه های نرم افزاری استفاده می شود . میزان خطا و دلایل آنها در یک پروژه نرم افزاری به میزان دقت مهندسین سازنده در گروههای کاری با استفاده از منشأ ارتباط استانداردی با یک دلیل منطقی در هنگام ساخت نرم افزار است این قضیه در مهندسین جوان بیشتر صدق می کند که آنها باید زیر نظر مهندسین با سابقه تری کار کنند تا هم به روش کار آشنا شده و هم خصوصیات نرم افزاری را بهتر درک کنند استاندارد ANSI با استفاده از نظرات صاحب نظران در قضیه میزان خطا و دلایل آنها یک پارادایم چند منظوره با استفاده از ۲ استاندارد IEEE و DIN ساخته است که در زیر آن را نشان می دهیم .



(شکل ۱۲)

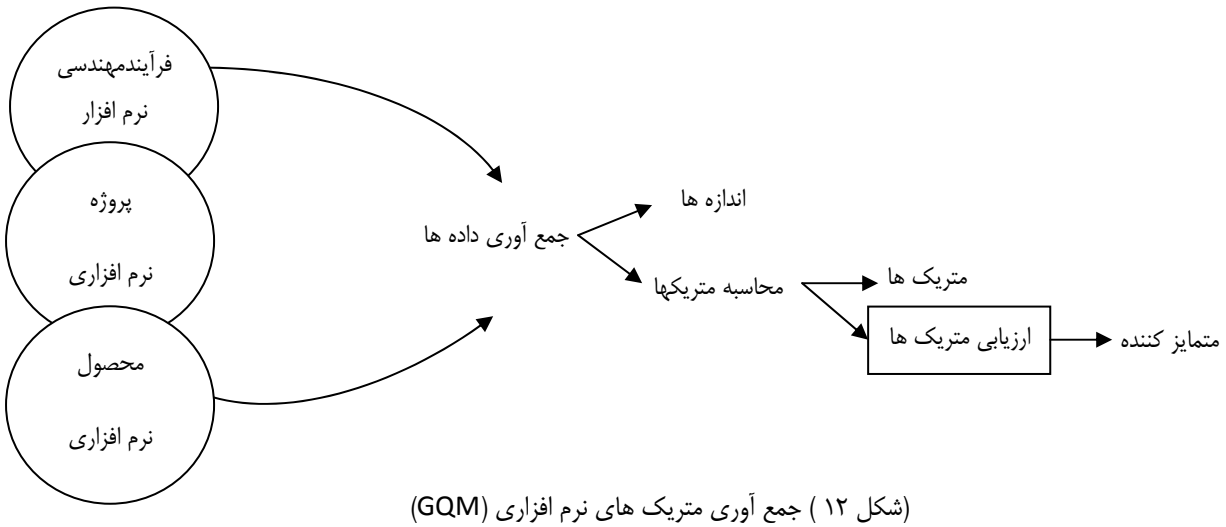
با نگاه کردن به شکل متوجه می شوید که رابط های نرم افزاری در مقدار کمترین آن قرار دارد . (دلایل عیوب و منشأ خطاها در سال ۹۹ در سازمان NASA با استفاده از روش تکمیلی GOM مورد استفاده قرار گرفت که بعداً به آن خواهیم پرداخت .)
حال با دانستن دلایل عیوب و منشأهای آنها سعی بر کم کردن خطاها و متریکهای انسانی داریم که با استفاده از تعداد خروجی ها اندازه گیری نرم افزار و متریکهای مبتنی بر size آنها را به بهترین صورت مدیریت کرده و رفع عیب می کنیم .

بررسی الگوی رفتاری GQM در متریک ها :

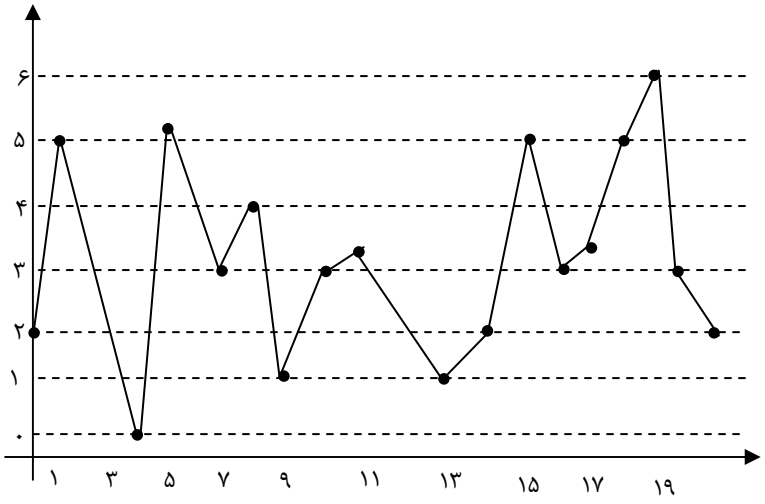
جهت سهولت اندازه گیری کارکرد نرم افزار در دهه ۹۰ اندازه گیری ها و استانداردهای زیادی استفاده می شد . ویکتور وایزلی از مهندسین سازمان ناسا در سال ۱۹۹۵ اقدام به توزیع روش GQM جهت اندازه گیری های متریک نرم افزار کرد . این الگو بر پایه تشخیص خطا در موقعیت های گوناگون بنا شده که توانسته است یک سازمان را در اندازه گیری صحیح موقعیت کار نرم افزار به پیش برد . (در حقیقت یک الگوی پیشرفت کیفیت را تکمیل نموده که GQM به خوبی در آن الگو جا می گیرد و این الگو شامل سه مرحله است .)

- ✓ فرآیند انجام برآورد در مورد یک پروژه .انتخاب ابزارهای مناسب و روشهای مدیریت فناوری برای رسیدن به هدف .
- ✓ فرآیند اجرای یک پروژه و نظارت بر داده های آن و همچنین عملکرد فرآیند داده ها توسط مدیر پروژه
- ✓ فرآیندی برای تحلیل داده ها و ارائه پیشنهادات جهت پیشرفت که مستلزم جستجوی مشکلات و داشتن الگویی مناسب از ساختار آنها ست.

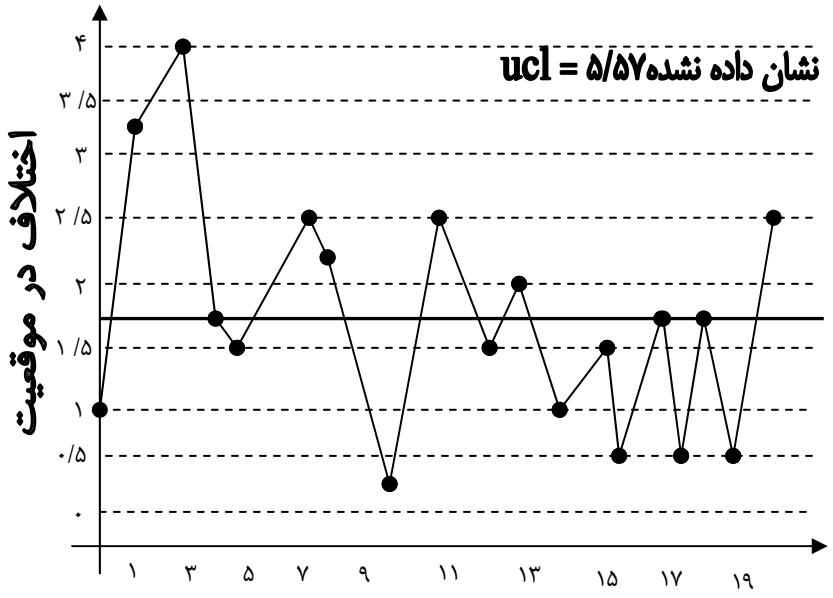
بایزلی توانست مجموعه ای از روشهای پیشنهاد شده را برای توسعه متریکها در پروژه های نرم افزاری مورد استفاده قرار دهد مجموعه اهداف برای ساختن چیزی که تحلیل شده است و استانداردهای بکار رفته را با استفاده از ۳ چارت عمده که شامل داده های متریک در خطاهای پوشش داده نشده و کنترل رنج حرکت در رسیدن به هدف و همچنین چارت کنترل معیار از معیار در خطاهای یافت شده در طی ساخت نرم افزار می باشد . GQM یکی از الگوهای زیر مجموعه ساخت یافته GQM با استفاده از معیار انحراف از پروژه در مسیرهای گوناگون در شرایط ساخت یکسان در نرم افزار می باشد حال به چارت های زیر نگاه کنید .



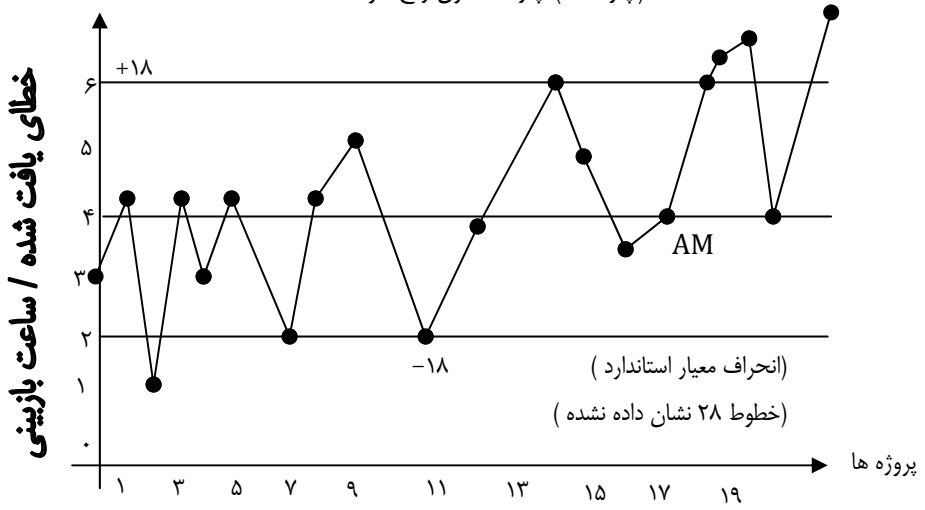
(شکل ۱۲) جمع آوری متریک های نرم افزاری (GQM)



(چارت ۱) داده های متریک برای خطاهای پوشش داده شده در هر ساعت بازبینی



(چارت ۲) چارت کنترل رنج حرکت



(چارت ۳) چارت کنترل

Ucl : این میانگین در حقیقت عددی است که باید اصل میانگین را در ucl مبنا که در هر چارتی متفاوت است ضرب نماییم و این حد را به عنوان حد بالای کنترل در نظر بگیریم . در GQM این مقدار ۵ برابر (به اندازه ۵ انحراف) بالای خط مرکزی قرار دارد .

تخمین منابع :

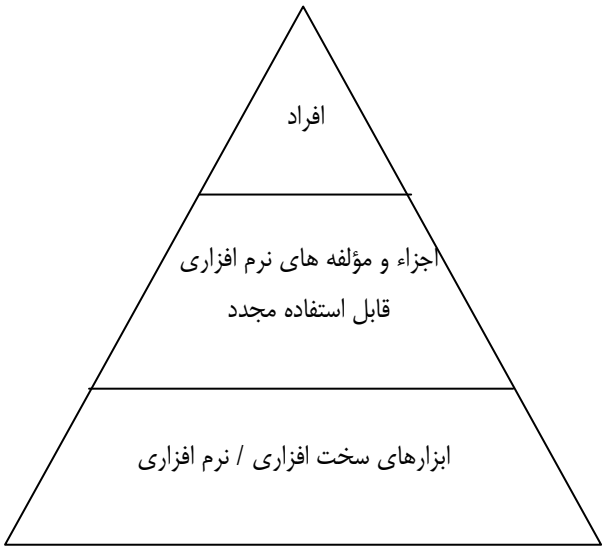
یکی از مهمترین برنامه ریزی های نرم افزاری تخمین منابع لازم در نیل به کار تولید نرم افزار است که در حقیقت با استفاده از محیط تولید و همچنین اجرای نرم افزاری بلوکهای تولید مشخص می شود . استاندارد EFQM منابع پروژه را در یک هرم سه مرحله ای به صورت بسط یافته توضیح می دهد که در بالای هرم افراد و در پائین ترین قسمت هرم ابزارهای سخت افزاری و نرم افزاری ساخت وجود دارد . در میانه هرم نیز اجزاء و مؤلفه های قابل بازیابی مجدد وجود دارد . EFQM توانسته است طرح ریزی کار خور را با استفاده از ارزیابی دامنه و انتخاب مهارتهای لازم در منابع انسانی شروع کرده با استفاده از موقعیت ها و پستهای سازمانی با استفاده از ارتباطات میان سازمانی بین رئیس پروژه و صاحب نظران پروژه یک منبع انسانی به صورت بی حد و مدام از انواع تخصص ها در دسترس داشته باشد .

مبحث منابع نرم افزاری با قابلیت استفاده مجدد :

همانطور که می دانید مهندسی نرم افزار بر قابلیت استفاده از کاربر مجدد دستورات و بلوکهای ساختمان نرم افزاری بصورت انواع کوپرها و همچنین بلوکهای جزء به جزء و قابل اجرا به خاطر سهولت در مرجع بندی و همجنین استانداردسازی در ادغام دستورات با استفاده از ۴ گروه منابع انسانی ، منابع نرم افزاری را بیان می کند که شامل :

۱. اجزای ساخته شده و آماده
۲. اجزایی که دارای تجربه کامل در مورد آنها هستیم .
۳. اجزایی که دارای تجربه نسبی در مورد آنها هستیم .
۴. اجزای جدید ابزارهای سخت افزاری و نرم افزاری ما در استاندارد EFQM بصورت بهینه با استفاده از مدل W5HH

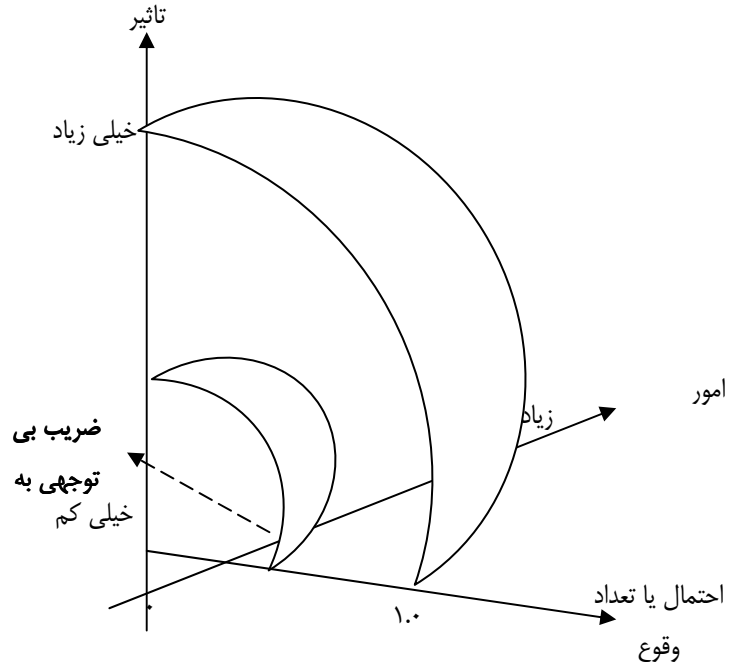
در نحوه تولید نرم افزار به ویژه استفاده از سخت افزارهای با قابلیت تطابق خودکار با نرم افزار در استفاده چند کاربرده برای بهینه کردن قابلیت تکامل نرم افزار برای رسیدن به نقطه عطف نظریه ریشه نرم افزار در بکارگیری بهتر نرم افزار و سخت افزار با سرعت قابل تطبیق با نرم افزار را ارائه کنند . به دیاگرام منابع پروژه در استاندارد EFQM نگاه کنید :



(شکل ۱۳) دیاگرام منابع نرم افزاری در EFQM

ارزیابی میزان اثر ریسک در مهندسی نرم افزار :

استاندارد ANSI ۳ عامل مهم ماهیت ، حوزه و زمان را از پیامدهایی می داند که در هنگام انجام ریسک آنها را دچار مشکل می کند ماهیت ریسک نشان دهنده نوع و مقدار خطر آن در هنگام Run کردن نرم افزار است که این ماهیت از مقدار ریسک کم شروع شده و به مقدار زیاد می رسد . باید توجه داشت که انجام ریسک در ساخت نرم افزار در پایه ماهیت آن برای شرکت نرم افزاری بسیار گران تمام می شود چون هرگونه اشکال پذیری در هنگام کارکردن برنامه کل سیستم را زیر سوال برده و فروش آن شرکت را تحت تاثیر قرار می دهد . در مورد حوزه ریسک پذیری نرم افزار باید گفت که این حوزه به انواع داخلی و خارجی تقسیم بندی می شود که ماهیت تاثیر پذیری نرم افزار در هنگام کارکرد و یا از طریق استفاده کننده آن مورد بررسی قرار می گیرد . باید دید که این مقدار ریسک چه مقدار به صورت جدی بوده و کارکردن نرم افزار را چگونه تحت تاثیر قرار می دهد . یک مدیر پروژه خواهان این است که زمان تاثیر پذیری نرم افزار از ریسک در زمان آزمایشی آن بدست آید و هرگز تاثیرپذیری زمان به مدتهای طولانی واگذار نشود نیروی هوایی آمریکا طرح RMMM را در مورد ریسک پذیری شناخت نرم افزار در سال ۹۲ ارائه کرده که با استفاده از فرمول اصلی Re=P.C به دست می آید . در این فرمول Re در معرض ریسک قرار گرفتن نام دارد . P بیانگر احتمال وقوع یک ریسک و C بیانگر هزینه وارده به پروژه است . همچنین این سازمان احتمال وقوع ریسک را در انجام امور مدیریتی به صورت یک دیاگرام توجیهی بیان کرده است که در آن احتمال وقوع ریسک توسط امور مدیریتی با توجه به ضریب بی توجهی به ریسک بیان می شود . به دیاگرام نگاه کنید :



(شکل ۱۴)

بررسی طرح RMMM : (تخفیف و نظارت مدیریت ریسک)

این طرح یک راهبرد مدیریت ریسک است که به صورت یک طرح تعدیلی به وسیله مدیریت سازمان انجام می شود . **این طرح از تمام کارهایی که به عنوان بخشی از تحلیل ریسک انجام می شود پشتیبانی می شود و توسط مدیر پروژه گزارش سازی و مستندسازی می شود.** در این طرح تیمهای نرم افزاری یک گزارش نرم افزاری تهیه کرد که و هر دیسک را بطور جداگانه با استفاده از صفحات اطلاعات ریسک RIS تهیه می کند و بر حسب ورود اطلاعات و اولویت ریسک طبقه بندی کرده به مدیر سازمان ارائه می کند کنترل فرآیندی که مدیر انجام می دهد دارای ۳ هدف اصلی است :

۱. بررسی اینکه آیا ریسکهای پیش بینی شده به هدف خواهند یافت ؟
۲. مراحل ناسازگاری ریسک که برای آن تعیین شده مناسب است یا خیر ؟

۳. جمع آوری اطلاعاتی که از آنها بتوان برای تحلیل بیشتر در ریسک استفاده کرد .

بطور نمونه یک صفحه RIS در اینجا گنجانده شده که در اینجا می بینید :

Risk ID	Date	Prob: %	Impact
Description:			
Refinment / Context:			
Mitigation / Monitoring:			
Management / Contingency Plan / Tigger:			
Curent Status:			
Originator :		Assigned:	

هزینه کیفیت :

مبحث هزینه کیفیت شامل تمام هزینه هایی است که در جهت نیل به کیفیت اجرایی بالا فراهم می شود و ایجاد بستری برای هزینه های فعلی و شناسایی فرصتهایی برای رسیدن به کیفیتهایی بالاتر در تولید مصنوعات نرم افزاری است برای ارزش یابی فرصتهایی که در رسیدن به بهبود پیشرفت مورد نیاز است یکسری هزینه ها که شامل هزینه های طراحی ، برنامه نویسی ، عملیاتهای میدانی و آزمون سیستماتیک نرم افزار است را شامل می شود .

نکته:

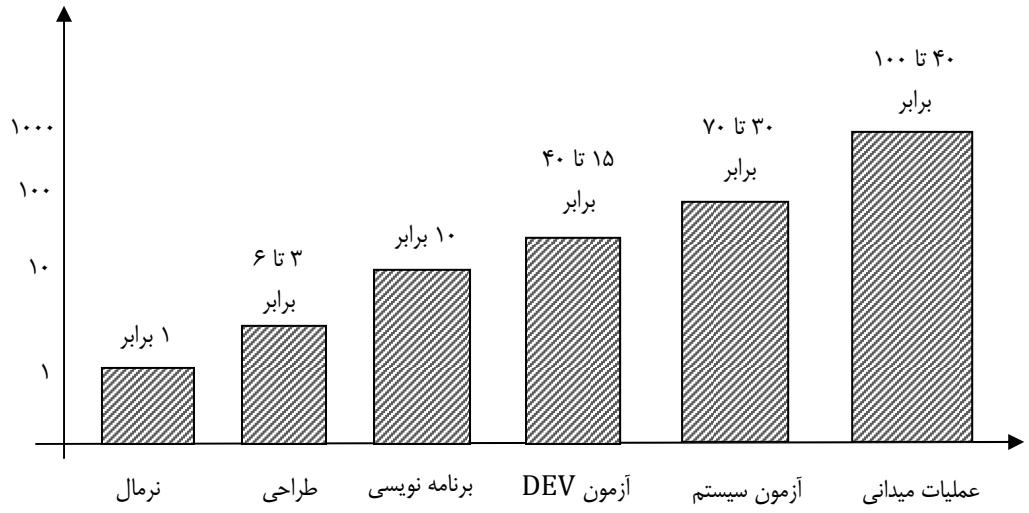
هزینه های کیفیت ممکن است به هزینه های مربوط به پیشگیری و ارزیابی شکست نیز ختم شود که در این سیستم هزینه های پیشگیری شامل :

✓برنامه ریزی های کیفیت ✓بررسی های منظم فنی

✓تجهیزات آزمون برای نرم افزار ✓آموزش به استفاده کنندگان است.

هزینه های ارزیابی شامل : فعالیتهایی برای آگاهی از وضعیت تولید در آغاز هر فرآیند است که هزینه هایی شامل انواع بازرسی های درون مرحله ای ، تعمیر و نگهداری تجهیزات و انجام آزمون است . در مبحث هزینه های شکست باید دید در صورت معلوم نشدن نقص قبل از تحویل محصول به مشتری چه هزینه هایی به شرکت تحمیل می شود این هزینه ها ، هزینه های درونی و بیرونی هستند که هزینه های داخلی شامل دوباره کاری ، تعمیرات و تجزیه و تحلیل وضعیت شکست است و هزینه های شکست خارجی مربوط به نقایصی بعد از ارسال محصول به مشتری می باشد که شامل هزینه های دعوای حقوقی بازگشت تحویل کالا ، پشتیبانی و کمکهای حمایتی و انجام تعهدات تضمین شده .

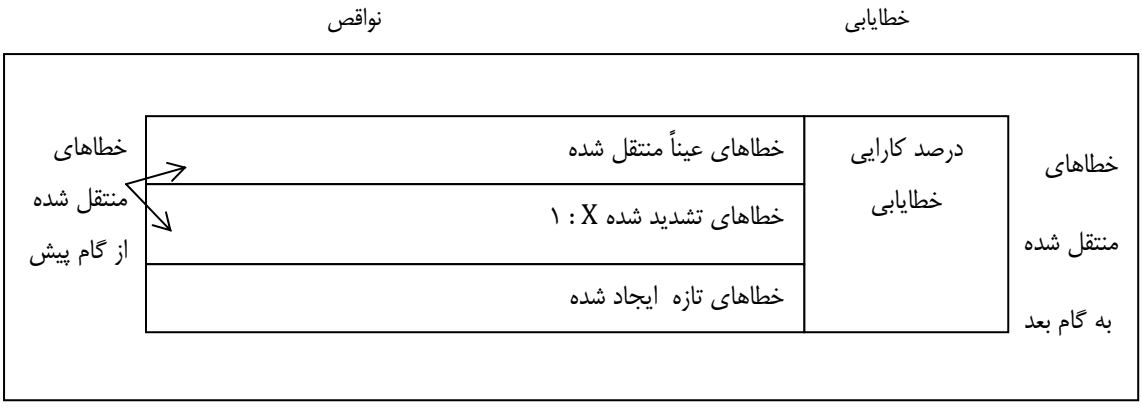
دیاگرام زیر دیاگرام هزینه های نسبی اصلاح یک خطا در استاندارد ANSI است که با استفاده از نظرات شرکت ماکروسافت در نرم افزار office 2003 انجام شده است . در حال حاضر TQM استاندارد مربوط به کیفیت است که در حال بارگذاری می باشد که بعداً راجع به آن صحبت می کنیم .



(شکل ۱۵)

تأثیر عیوب بر هزینه های نرم افزاری :

ااستاندارد IEEE واژه نقص را به عنوان مهمترین عیب نرم افزاری می شناسد همچنین در یک نرم افزار پیدا کردن عیب قبل از تحویل به بازار مهمترین وظیفه گروه نرم افزاری است در صورت وجود نقص گروه نرم افزاری شرکت را دچار ضرر کرده و در فرایند فروش خلل ایجاد می کند در حقیقت یافتن خطا در نرم افزار و رفع عیب آن مهمترین وظیفه گروه نرم افزاری است که اگر به هنگام اجرا نشود عرضه نرم افزار با مشکل روبرو شده و فروش آن پائین می آید در صورت کشف زود هنگام خطاها (نوشته شده در مجله نیپون الکتریک NIPPE) به صورتی که خطاها نتواند در مرحله بعدی ساخت نرم افزار نشر یابند و اگر تعداد آنها از ۵۰ تا ۶۵ درصد خطاها بیشتر باشد استاندارد IEEE استفاده از اصل نرم افزار را برای عرضه مجاز نمی دانند برای نشان دادن اثر هزینه ها در نرم افزار و ردیابی زود هنگام خطاها ما یکسری از هزینه های نسبی را که براساس داده های واقعی از کار پروژه های نرم افزاری جمع آوری کرده و مورد مطالعه قرار می دهیم در ردیابی خطاها از یک Datagram به شکل زیر استفاده می کنیم که به نام مدل تشدید توسعه نقص یابی است . این مدل از سال ۹۷ به بازار عرضه شده و دارای خطاهای منطبق شده از گام پیش به روشهای مختلف و پیدا کردن درصد کارایی خطایابی در کل نرم افزار است به Datagram نگاه کنید :



(شکل ۱۶) پارادایم تشدید

رهیافتهای رسمی برای تضمین کیفیت نرم افزار :

همانطور که می دانید کیفیت نرم افزار و مقدار آن بر عهده همه افراد گروه است و در کل پروسه طراحی ، ساخت و آزمایش گروه باید به صورت متشکل در امر پیشبرد ساخت پروژه نرم افزاری کمک کند .

استفاده از استانداردها ، شاخص ها و انواع متریک ها در بالا بردن ضریب کاری نرم افزار بسیار مهم بوده و برای یک گروه نرم افزاری ساخت یک نرم افزار همانند یک شی ریاضی بوده (مسئله ملموس) که با رهیافتهای توسعه یافته و با استفاده از انواع سیستمهای کاربرد بهینه نظریه ها به بهترین کیفیت نرم افزاری برسیم زیرا تلاش و تعیین صحت برنامه از سمت گروه نرم افزاری باعث بالا بردن ضریب اطمینان آن و کیفیت نرم افزار می شود . ویلیام دایکستر در دانشگاه میشیگان در سال ۸۴ موفق به تعیین صحت و تضمین برنامه های نرم افزاری با استفاده از مفاهیم ساخت یافته شد که در آن تضمین کیفیت انجام شد که در یک نرم افزار نمونه (SQA) و تحلیل داده های مهم و بررسی تضمین کیفیت آن گردید به جدول نمونه دایکستر مورد استفاده در نرم افزار SQA نگاه کنید در این جدول خطاهای MCC ، IBS ، EDR به عنوان دلایل اساسی انتخاب شده در حالیکه ۵۳٪ تمام خطاها را تشخیص می دهند خود دلایل اساسی هستند اما یک سازمان مهندسی نرم افزاری برای بهبود EDR و استفاده از مدل سازی های ابزارهای موردی (case) استفاده می کنند و ا ین نکته در بررسی جدول حائز اهمیت است که انواع فعالیتهای اصلاحی در **ابتدا و وهله نخست بر علل اساسی متمرکز است و با تصحیح آن مسائل ریزتر نمایان می شود .** به جدول نگاه کنید :

توضیح : خطاهای نام برده شده در فایل های مهم SQA موجود بوده و ممکن است در هر نرم افزاری متفاوت باشد .

نوع خطا	کل		مهم		متوسط		جزئی	
	تعداد	درصد	تعداد	درصد	تعداد	درصد	تعداد	درصد
IES	۲۰۵	٪۲۲	۳۴	٪۲۷	۶۸	٪۱۸	۱۰۳	٪۲۴
MCC	۱۵۶	٪۱۷	۱۲	٪۹	۶۸	٪۱۸	۷۶	٪۱۷
IDS	۴۸	٪۵	۱	٪۰/۸۶	۲۴	٪۶	۲۳	٪۵
VPS	۲۵	٪۳	۰	٪۰	۱۵	٪۴	۱۰	٪۲
EDR	۱۳۰	٪۱۴	۲۶	٪۲۰	۶۸	٪۱۸	۳۶	٪۸
ICI	۵۸	٪۶	۹	٪۷	۱۸	٪۵	۳۱	٪۷
EDL	۴۵	٪۵	۱۴	٪۱۱	۱۲	٪۳	۱۹	٪۴
IET	۹۵	٪۱۰	۱۲	٪۹	۳۵	٪۹	۴۸	٪۱۱
IID	۳۶	٪۴	۲	٪۱/۶۶	۲۰	٪۵	۱۴	٪۳
PLT	۶۰	٪۶	۱۵	٪۱۲	۱۹	٪۴/۶	۲۶	٪۶
HCI	۲۸	٪۳	۳	٪۲	۱۷	٪۴	۸	٪۲
MIS	۵۶	٪۶	۰	٪۰	۱۵	٪۲/۳	۴۱	٪۹
جمع کل	۹۴۲	٪۱۰۰	۱۲۸	٪۱۰۰	۳۷۹	٪۱۰۰	۴۳۵	٪۱۰۰

بررسی کیفیت استانداردهای سری ISO در مهندسی نرم افزار:

هدف از این مبحث توصیف استاندارد بین المللی ISO می باشد این استاندارد که توسط ۱۳۰ کشور دنیا پذیرفته شده بصورت مستمر بر اهمیت آن به عنوان ابزاری که به وسیله آن مشتری می تواند **توانایی سازندگان نرم افزار را مورد قضاوت قرار دهد افزوده شده است .** مشکل اصلی در این استاندارد مربوط کردن استاندارد سری ISO به صفتی خاص است که با اصطلاحات عمومی بیان می شود نخست معنی استاندارد ISO9001 را بررسی می کنیم

برای صفت نرم افزاری استانداردهای مربوط عبارتند از :

۱) **ISO9001** : مدلی برای تضمین کیفیت در طراحی ، توسعه ، تولید و نصب نرم افزار می باشد .**این استانداردی است که سیستم کیفیت بکاربرده شده برای تضمین توسعه محصول از نوع نرم افزاری را توصیف می کند .**

۲) **ISO9000 :** رهنمودهایی است که برای بکارگیری ISO9001 در توسعه و نگهداری نرم افزار و سال ۲۰۰۰ این ISO برای تولیدکنندگان نرم افزار اجباری شده است .

۳) **ISO9004 :** این استاندارد **مدیریت کیفیت و عناصر سیستم کیفیت را به همراه خدمات و تحصیلات نرم افزاری چون حمایت از مصرف کنندگان فراهم می کند .** نیازمندیهای ISO9004 به قرار زیر است :

- ✓ تجهیزات آزمون و اندازه گیری بازرسی
- ✓ بازرسی و وضعیت آزمون
- ✓ اقدامات اصلاحی
- ✓ کنترل بر محصول ناسازگار
- ✓ ثبت کیفیت
- ✓ بازرسی درون کیفیتی
- ✓ و استفاده از فنون آماری

تجهیزات و بررسی هایی که بالا یاد کردیم با استفاده از انواع زیرمجموعه های EFQM , EQS , TS در ساختار زیرمجموعه استاندارد ISO قابل بررسی بوده و با استفاده از سلسله مراتب بازرسی دوره ای نرم افزار (مخصوصاً در نسخه Beta) کارایی ویژه سیستماتیک یک نرم افزار را در بالا بردن سیستم کارایی نرم افزار به ویزه کم کردن خطاها در مرحله اجرای نرم افزار یاری می کند .

نکته :

برای اطلاعات بیشتر به سایت WWW. ISO.COM بروید.

مدیریت پیکربندی نرم افزار :

به آن SCM می گویند . استاندارد ANSI خروجی فرآیند نرم افزاری را شامل ۳ سری از اطلاعات دانسته است که :

۱) برنامه های کامپیوتری در سطح منبع و فرم های قابل اجرا

۲) اسنادی که برنامه ها را شرح می دهند .

۳) داده ها با استفاده از اطلاعات بالا قسمت پیکربندی نرم افزار حاصل می شود .

با پیشرفت پروژه نرم افزرای وضعیت نرم افزرای که به آن SCIS می گویند افزایش یافته و با استفاده از یک منشأ در طرح نرم افزار در SCI ممکن است سردرگمی حاصل شده و تغییرات زیاد شود . طرحی را طبق استاندارد ANSI ۴ منبع اساسی دارد که:

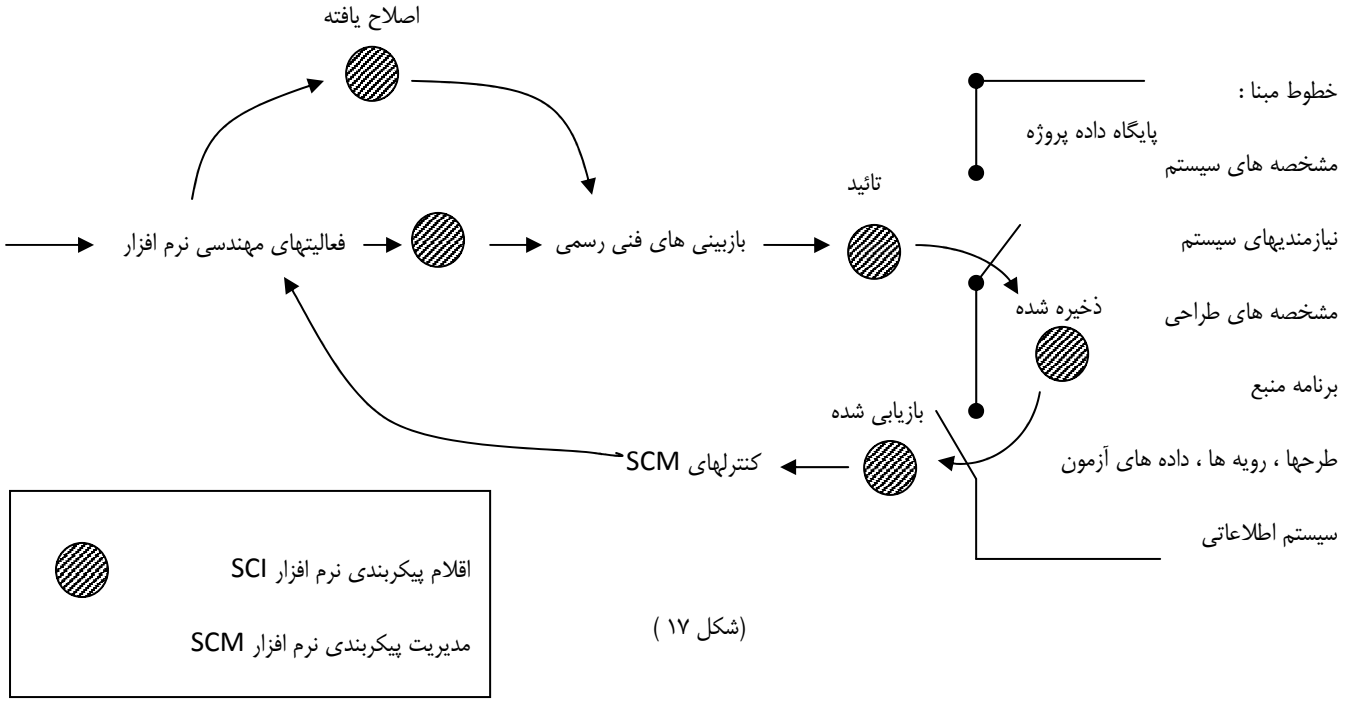
- ✓ شرایط جدید کاری و تجاری که به ما دیکته می شود
- ✓ نیازهای جدید مشتریان که باعث عملکرد نرم افزار شده و خدمات کامپیوتری برای آن نرم افزار ارائه می شود .
- ✓ سازماندهی های دوباره که در ساختار تیم نرم افزار به وجود می آید .
- ✓ مشکلات مربوط به بودجه و برنامه ریزی که ممکن است سبب دوباره تعریف شدن محصول شود .

مدیریت پیکربندی نرم افزار **SCI طبق تعریف عبارتست از مجموعه فعالیتهایی که برای کنترل تغییر در طول چرخه روند و زندگی نرم افزار به وقوع می پیوندد** . برای استنباط بهتر SCM و SCI احتیاج به یک خط مبنا داریم که در مفهوم مدیریت وضعیت نرم افزرا به ما کمک کند .

استاندارد IEEE با شماره ۶۱۰/۱۲ که در سال ۱۹۹۰ نوشته شده خط مبنا را این گونه تعریف می کند :

خط مبنا :

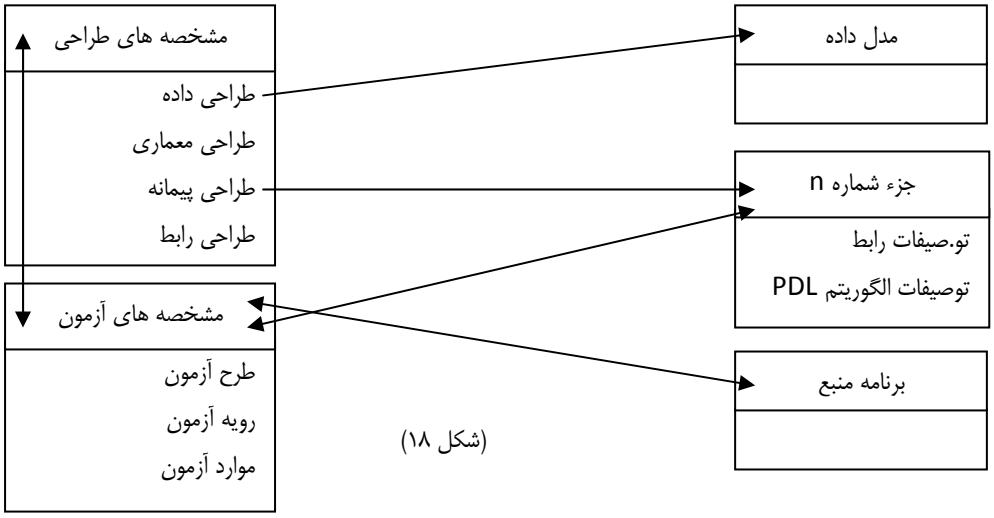
یک ویژگی و یا محصولی که به طور اساسی مورد بررسی قرار گرفته و به تایید رسیده باشد پس از آن به عنوان پایه و اساس مراحل بعدی پروژه مورد استفاده قرار می گیرد و تغییر آن به واسطه استفاده از شیوه های اساسی است شکل زیر بیان کننده دیاگرام SCI های خط مبنا و پایگاه داده ای پروژه است که با استفاده از آن روش کار با SCI مبنا به راحتی روشن می شود .



(شکل ۱۷)

اقلام پیکربندی نرم افزار :

برای پیکربندی نرم افزار از یک سری ویژگی های آن باید استفاده کرد و استفاده از SCI به عنوان سندی که مجموعه کاملی از موارد آزمون را برای یک نرم افزار داشته باشد استفاده می کنیم در واقع SCI برای تشکیل اشیای پیکربندی سازمانی استفاده می شود و از طریق یک سری روابط شی گرا و ویژگی های طراحی و رابطه های ترکیبی سیستم پیکربندی شده و کار می کند . الگوها جزء n بخشی از طراحی هستند که با استفاده از ویژگی های مدل داده ها و برنامه های منابع تحت تاثیر قرار گرفته و پیکربندی می شود در حقیقت برای یک تابع در C++ دیاگرام زیر صادق بوده و مشخصه های آزمون در طرح ، رویه و مؤثر بودن آن تحت تأثیر مشخصه های طراحی است (به صورت مستقیم) با توجه به دیاگرام پیکربندی اشیا در زبان C++ این گونه استنباط می شود که **ویژگی طراحی مدل داده ها جز حرف n برای سایر منابع مشخص آزمون بطور جداگانه قابل تعریف است .** به دیاگرام نگاه کنید :



(شکل ۱۸)

فرآیند مدیریت پیکربندی نرم افزار :

مدیریت پیکربندی نرم افزار رکن مهمی از تضمین کیفیت نرم افزار است که مسئولیت اولیه آن کنترل در تغییر می باشد اما به عنوان نمونه اقلام پیکربندی نرم افزار SCI و همچنین مدیریت پیکربندی نرم افزار به طور مرتبط و مدام گزارش تمام تغییرات اعمال شده را به پیکربندی انجام می دهد چون راجع به SCM بحث کردیم باید یکسری سؤالات را مورد بررسی قرار دهیم و طبق نظر استاندارد ANSI سؤالات اینگونه هستند :

۱. یک سازمان چگونه می تواند نسخه های یک برنامه را بگونه ای درست شناسایی و کنترل نماید ؟
۲. یک سازمان چگونه می تواند تغییرات را قبل از در اختیار قرار گرفتن نرم افزار نزد مشتری کنترل نماید ؟
۳. چه کسی مسئول اولویت بندی های تغییرات است ؟
۴. برای ارزیابی تغییرات از چه مکانیزمی استفاده می کنیم ؟

این سؤالات ما را بر آن می دارد که **۵ کار اصلی SCM را تعریف کنیم که شناسایی ، کنترل نسخه ها ، کنترل تغییر ، بررسی پیکربندی و گزارش دادن می باشد .**

شناسایی اشیا پیکربندی نرم افزار :

برای کنترل و اداره پیکربندی نرم افزار آنها را باید به صورت جداگانه و با اسفاده از یک رهیافت شی گرا سازماندهی کرد طبق نظر استاندارد ANSI دو نوع شی قابل بررسی است که یکی از نوع شی های پایه و دیگری شی های مجتمع است که شی پایه عبارت است از واحد متن که توسط یک مهندس نرم افزار در هنگام کدنویسی با استفاده از داده های اولیه نرم افزار و استفاده از بارگذاری جامع و نگاه کردن به خصوصیات نیازمندیهای نرم افزار برنامه نویسی کرد و با استفاده از مدل داده های جزئی و استفاده از گراف تکامل طبق نظر استاندارد ISO با استفاده از یک تحقیق شی گرا یک رشته کاراکترها را شناسایی کرده و با استفاده از یک توصیف مستمر پایه به شناسایی مواردی که در زیر گفته می شود اقدام می کنیم .

- ✓ سند برنامه های داده ای را مشخص می کنیم .
- ✓ به شناسایی پروژه می پردازیم .
- ✓ اطلاعات مربوط به نسخه و یا ورژن نرم افزار را بررسی می کنیم .

می دانید که منابع ما محدودیتهایی هستند که ارائه شدند پردازش گشته و ارجاع شدند . تمامی این اقدامها توسط شی های پایه انجام شود که در هنگام استفاده مهندس نرم افزار از آنها در شناسایی پیکربندی سیستماتیک نرم افزار روابطی درون گرا بر پایه ساختار شی گرا برای رسیدن به کمترین اشکال و استفاده مستمر از حافظه heap بلوک های منطقی حافظه بصورتی شی گراست برای بررسی خصوصیات طراحی شی و یک سلسله مراتب (به عنوان مثال در SCI) در طول مسیرهای مختلف دادن شماره به شی برای رسیدن به مسیرهای سیستم و در مواردی تحلیل ساختار شی گرای آن بسیار مفید است .

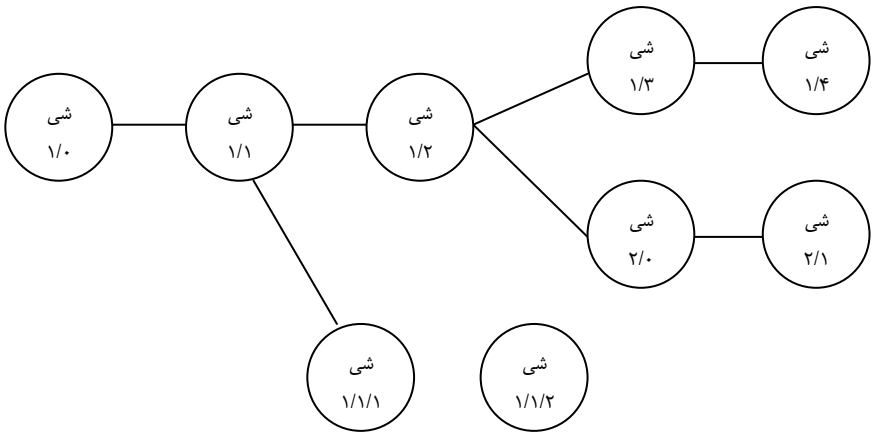
نکته :

درآنالیز گراف شی گرای ما شاخه های خط اصلی باید بصورت متقارن و هم وزن در هر دو طرف شاخه اصلی وجود داشته باشند . چنانچه گراف توازن ما دارای شاخه های بسط یافته غیر متعارف باشد سیستم بصورت غیر واقعی پایه گذاری شده و اقلام پیکربندی نرم افزار ما در رسیدن به واحد شی گرا دچار مشکل است .

توجه :

استاندارد ANSI دو مفهوم شی گرا و شی متراکم و مرکب را بصورت بازنمایی از نسخه کامل یک پیکربندی نرم افزار محسوب می کند .

به گراف تکامل نرم افزار SCI نگاه کنید :



(شکل ۱۹) گراف تکامل شی گرایی

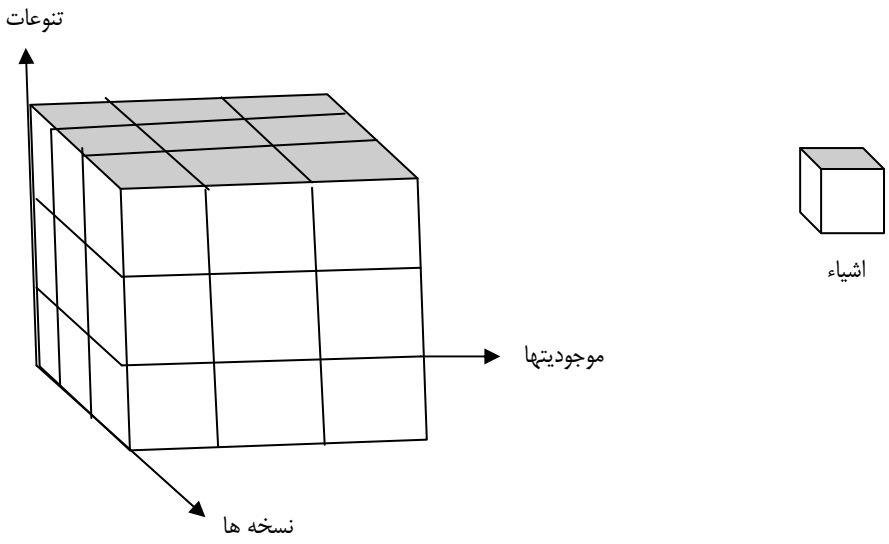
در آنالیز گراف شی گرایی یک سری روابط که در حقیقت از نمودار ۱/۱ شروع شده و به نمودار ۱/۴ ختم می شود بر می خوریم این مدل با بررسی انواع ER یک مدل جریان داده ایی بصورت موزون به دست می دهد و با استفاده از بر هم بندی نشانه های سیستم های شی گرا کمک می کند تا هر نسخه از روی یک سیستم خودکار ساخته شود .

در شناسایی شی های نرم افزار باید دقت کنیم که طول شیی نرم افزار هر بار چند بار تغییر می کند . تغییر طول شیی نرم افزار با استفاده از مدل گراف تکامل آن با بررسی Base مدل سازگار آن در رسیدن به بهترین بهره برای انجام و بارگذاری درست نرم افزار و پشتیبانی از نسخه های قبلی انجام می پذیرد . این امکان وجود دارد که هر نسخه از نرم افزار دچار تغییر شود اما نه الزاماً همه نسخه های آن تغییرات

اعمال شده بر روی طرح مستندسازی نسخه جامع نرم افزار در بررسی مجموعه های شی گرای نرم افزار در رسیدن به حالت پایدار با استفاده از گراف تکامل نرم افزار که سیستم را در هر بار بررسی ملزم به بررسی طول گراف آن می نماید انجام داد . مجموعه های متناظر نرم افزار که در منبع و اسناد و داده ها ترکیبی اختلاطی از مجموعه های زمان گریز موجود در تک تک نسخه های نرم افزار استفاده می شود و استاندارد ANSI پیکربندی سیستم ها را از طریق ربط دادن صفتهای هر نرم افزار با نسخه های بعدی از طریق توصیف مجموعه های دلخواه پشتیبانی می کند . (پشتیبانی واژه موجودیت نهادها به تمام اشیای ترکیبی و ساده که در قلمرو پیکربندی نرم افزار را وجود دارد اطلاق می گردد .)

کنترل نسخه های نرم افزار :

در این روش کنترلی به روشها و ابزارهایی با هم ترکیب شده تا نسخه های گوناگونی از شی ها با پیکربندی های مختلف را طی فرآیند ساخت نرم افزار که ایجاد شده مهار کنیم . کنترل نسخه را در یک متن اینگونه توصیف می کنند که این پیکربندی به کاربر امکان می دهد تا گزینه های پیکربندی گوناگونی از سیستم نرم افزاری به طریق انتخاب نسخه های مناسب مشخص کند . این امر از طریق ربط دادن صفات با هر یک از نسخه های نرم افزاری با پیکربندی از طریق توصیف مجموعه پشتیبانی می شود . این صفات به سادگی در یک نسخه ی خاص موجود هستند . اما می توان این صفات را به دسته های مختلف تقسیم کرد و هرکدام را در ورژنی خاص گذاشت . استاندارد ISO فهرستی از یک ویژگی های استاندارد که برای یک نرم افزار قابل اجرا است را ارائه کرده است به این مفهوم که رابطه ی بین موجودیتها متغیرها و اصطلاحات ما عبارت است از ارائه آنها به عنوان یک مخزن از اشیاء با مراجعه به جدول پائین رابطه ی بین شی های پیکربندی شده ، موجودیتها و متغیرها در نسخه های بعدی در یک فضای سه بعدی نشان داده شده که در این شکل یک موجودیت تشکیل شده است از **مجموعه ای از شی ها که نسخه های تجدید نظر شده ی آنها در یک سطح قرار گرفته و یک متغیر کجکوعه ای است بصورت متفاوت از شی ها که نسخه ی تجدید نظر شده ی آنها در یک سطح می باشد پس یک متغیر مجموعه متفاوتی از شی هاست که نسخه ی تجدید نظر شده ی آنها در یک سطح است پس در موازات سایر متغیرها قرار می گیرد . وقتی که تغییرات عمده بر روی یک یا چند شی صورت می گیرد یک نسخه ی جدید تعریف می شود .**



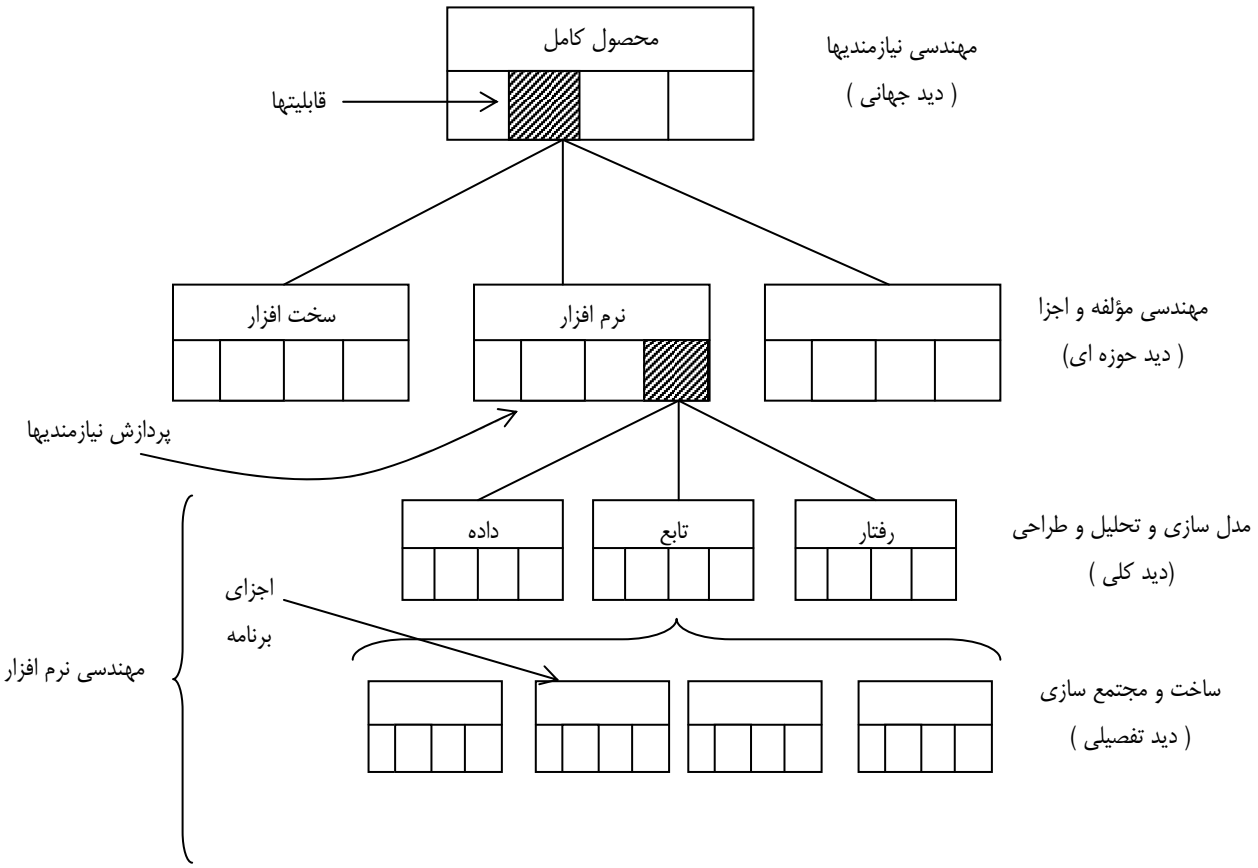
(شکل ۲۰) مخزن شی ، بازنمایی اشیاء تنوعات و نسخه ها (REI89)

مرجع استانداردی :

استانداردهای مدیریت پیکربندی نرم افزار طی ۲۰ سال تهیه شده که در ابتدا استانداردهای MIL-STD483 و DOD-STD480A و MIL-STD-1521A می بود که ظرف مدت ۱۰ سال با جهش هایی توسط دو استاندارد ANSI و IEEE به همراه ISO ، DIN با شماره های ۱۰۲۸ ، ۱۹۸۸ ، ۱۹۲۸ رسید . نرم افزارهای غیرنظامی با ۲ استاندارد ANSI و IEEE سروکار دارند که این استانداردها بر پایه اولیه در ساخت نرم افزار و استفاده از اشکال زدایی در نسخه Beta کار می کند . آشنایی مهندسان با استانداردهای ذکر شده به آنها کمک می کند تا در بارگذاری صحیح اطلاعات و در یافتن روشهای صحیح ساخت نرم افزار یاری نماید .

برداشتهایی از مهندسی محصول :

هدف از مهندسی محصول این است که خواسته های خریداران را در مقابل یک رشته از قابلیتهای نرم افزار به محصول تبدیل شود اصول و روشهای معماری زیر ساخت، ساخت نرم افزار طبق نظر استاندارد ANSI یک معماری ۴ پایه است . که پایه های اصلی آن عبارت است از : نرم افزار ، سخت افزار ، داده ها و نفرات با توجه به نمودار سلسله مراتبی محصول در استاندارد ANSI دریافتن اینکه نیازهای کنترل و عملکرد یک نرم افزار و همچنین دریافتن رفتار خاص آن با توجه به ورودی داده ها و انواع مؤلفه های اجزای مهندسی به وسیله یک مهندسی انسانی با استفاده از قابلیتها و تفاتتهای مهندسی در ساخت نرم افزار ممکن می شود . استاندارد ANSI یک سلسله مراتب مهندسی محصول را در سال ۲۰۰۳ بر روی نرم افزار OFFICE شرکت ماکروسافت اجرا کرده که به صورت یک پارادایم موضوعی با استفاده از مدل سازی و طراحی داده ها در یک Function جزئی و کلی برای رسیدن به بهترین محصول ارائه نموده است در این پارادایم تمامی المانها بر اساس جایگزینی مرتبط انتخاب نشده و رفتار کل نرم افزار به صورت بهینه مدنظر قرار گرفته است به پارادایم نگاه کنید :



(شکل ۲۱)

نمودار بافت سیستم یا SCD :

هر سیستمی می تواند به عنوان انتقال دهنده اطلاعات که در قالب ورودی ، پردازش و خروجی انجام می گیرد کارکند . سازمان NASA به عنوان پردازش رابط های کاربر و ویژگی های نرم افزار در تمامی سیستمهای مبتنی بر کامپیوتر اقدام به ساخت سیستمی الگویی با عنوان SCD کرده است که در آن سیستم مدل اجرای سیستم به گونه ای است که هرگام به جلو در حقیقت پایه ای برای گام بعدی است که در یک الگوی مهندسی با استفاده از پردازش رابط کاربر با محیط و دو زیر ساختار کارکردهای کنترل و نگهداری آن یک سیستم قالب برای مدلیزه کردن سیستمها در جهت ارتقای خروجی اطلاعات با استفاده از مقدار بهینه ورودی به نحوی که کارکرد سیستم به صورت یک نرم افزار با کمترین مقدار اشغال حافظه بالاترین راندمان را بدهد کار می کند . به قالب سیستم مدل مهندسی SCD در ساخت مدل مبنا نگاه می کنیم .

(شکل ۲۲) قالب مدل سیستم