فصل ۴ متریک‌های پروژه و فرآیند نرم‌افزار

مفاهیم کلیدی (مرتب بر حروف الفبا)
امتیازات کارکردی ، بهبود آماری فرآیند نرم افزار(SSPI) ، جمع آوری متریکها ، کارآیی رفع نقص ، کنترل فرآیند آماری ، متریک های پروژه ، متریک های فرآیند ، متریک های کیفیت ، متریک های مبتنی بر اندازه ، نتیجه گیری وارون

KEY CONCEPTS
Backfiring defect removal , efficiency , function points , metrics collection , project metrics , process metrics , quality metrics , size-oriented metrics , statistical process control , SSPI

نگاه اجمالی

متریک و اندازه‌گیری چیست؟ متریک های فرآیند و محصول نرم افزاری اندازه‌گیریهای کمیتی هستند که به افرادی که با نرم‌افزار کار می‌کنند امکان می‌دهند تا نسبت به کارآمد بودن فرآیند و پروژه‌های نرم افزاری بینش پیدا کنند. پروژه هایی که با استفاده از فرآیند به عنوان یک چهارچوب کاری، هدایت می شوند. داده‌های اولیه بهره‌وری و کیفیت جمع‌آوری می‌شوند. سپس این داده‌ها مورد تحلیل قرار گرفته و با میانگین ارقام گذشته مقایسه می‌شوند و برای تعیین بهبود و پیشرفت بهره وری و کیفیت مورد ارزیابی قرار می‌گیرند. همچنین متریک برای مشخص کردن نواحی و حیطه‌های مشکل آفرین بکار می‌روند بطوریکه راه حل هایی پیدا شود و فرآیند نرم‌افزاری بهبود یابد.

چه کسی این کار را انجام می دهد؟ متریک نرم‌افزاری توسط مدیران نرم‌افزاری مورد تحلیل و ارزیابی قرار می‌گیرند. این اندازه‌ها اغلب بوسیله مهندسان نرم‌افزار جمع‌آوری می شوند.

چرا این کار اهمیت دارد؟ اگر شما اندازه‌گیری نکنید، قضاوت شما فقط می‌تواند براساس ارزیابی ذهنی باشد. بوسیله اندازه‌گیری می‌توان روند را (چه خوب یا بد) دریافت و برآوردهای بهتری انجام داد و طی گذشت زمان بهبودها و پیشرفتهای واقعی اتفاق می‌افتند.

این مراحل و اقدامات کدامند؟ ما از تعریف یک مجموعه محدود از اندازه‌های محصول، پروژه و فرآیند که جمع‌آوری آنها ساده است شروع می‌کنیم. این اندازه‌ها اغلب با استفاده از متریک مبتنی بر اندازه یا کارکرد بصورت نرمال درمی‌آیند. نتیجه مورد تحیل قرار گرفته و برای پروژه‌های مشابهی که در سازمان اجرا می‌شوند با میانگین های گذشته مقایسه می‌شود. روندها مورد ارزیابی قرار گرفته و نتایجی بدست می‌آید.

حاصل کار چیست؟ مجموعه‌ای از متریک نرم‌افزاری که بینش کافی برای فرآیند و درک و فهم یک پروژه را فراهم می‌آورند.

چگونه مطمئن شوم که آن کار را درست انجام داده‌ام؟ از طریق بکارگیری یک طرح اندازه‌گیری منسجم و در عین حال ساده که هرگز برای ارزیابی، تقدیر یا توبیخ عملکرد شخصی بکار نمی‌رود.

اندازه‌گیری و سنجش یکی از مسائل ضروری برای هر یک از علوم و رشته‌های مهندسی است و مهندسی نرم‌افزار هم از این امر مستثنی نیست. اندازه‌گیری به ما امکان می‌دهد تا از طریق فراهم آوردن مکانیزمی برای ارزیابی هدفمند، به بینش و تفکر راه پیدا کنیم. لُرد کلوین (Lord Kelvin) زمانی گفته بود:

هنگامیکه شما می‌توانید چیزی را که درمورد آن صحبت می‌کنید، اندازه گیری نموده و بصورت اعداد بیان کنید، شما درمورد آن چیزهایی می‌دانید، ولی هنگامیکه نتوانید آنرا اندازه‌گیری کنید و هنگامیکه نتوانید آنرا درقالب اعداد بیان نمائید دانش شما درمورد آن چیز، بسیار ناقص و غیررضایتبخش است، این ممکنست نقطه آغاز دانش باشد ولی شما بندرت در تفکر و اندیشه‌تان پیشرفت می‌کنید و احتمال کمی وجود دارد که به مرحله علمی برسید.

جامعه مهندسی نرم‌افزار در نهایت سخنان لُرد کلوین را جدی گرفت. ولی این کار همراه با کمی ناامیدی و بحث و جدلهایی بود. سیستم متریک نرم‌افزار به دامنه وسیعی از اندازه‌گیریها برای نرم‌افزار کامپیوتر اشاره دارد. اندازه‌گیری را می‌توان با هدف بهبود فرآیند نرم افزاری براساس مستمر و

مداوم برای آن بکار گرفت. اندازه‌گیری را می‌توان از طریق یک پروژه نرم افزاری برای کمک به تخمین‌ها و برآوردها، کنترل کیفیت، ارزیابی بهره وری و کنترل پروژه بکار برد. در نهایت اینکه، اندازه‌گیری می‌تواند بوسیله مهندسی نرم‌افزار برای کمک به ارزیابی کیفیت محصولات کاری تکنیکی و برای کمک به تصمیم‌گیری تکنیکی در زمان پیشرفت یک پروژه مورد استفاده قرار گیرد.

متریک های فنی برای مهندسی نرم افزار، در فصل ۱۹ و ۲۴ ارائه شده اند.

در زمینه مدیریت پروژه نرم افزاری، ما در ابتدا به متریک کیفیت و بهره وری می‌پردازیم – اندازه‌گیریهای «خروجی» توسعه نرم‌افزار بعنوان یک عملکرد تلاش و زمانی که بکار گرفته شده و

اندازه‌گیریهای «متناسب کاربرد» محصولاتی که از کار حاصل آمده‌اند. پارک ، گافرت و فلوراک [PAR96][1] در کتاب راهنمایشان درمورد اندازه‌گیری نرم افزاری، دلایلی را ارائه کرده‌اند که چرا مسائل زیر را اندازه‌گیری می‌کنیم:

چهار دلیل برای اندازه‌گیری منابع، محصولات و فرآیندهای نرم‌افزاری عبارتند از:

- *مشخص کردن.*
- *ارزیابی کردن.*
- *پیش بینی کردن.*
- *پیشرفت کردن.*

ما برای رسیدن به فهم درک درمورد فرآیندها، محصولات، منابع و محیط‌ها، اندازه‌ها را مشخص می‌کنیم و برای تعیین خط مبنا برای مقایسه با ارزیابی‌های آینده این کار را انجام می‌دهیم.

ما برای تعیین وضعیتها باتوجه به طرح‌ها، ارزیابی می‌کنیم، اندازه‌ها سنسورها (حس‌کننده‌ها)یی هستند که به ما امکان می‌دهند بفهمیم چه موقع پروژه‌ها یا فرآیندهای ما از راه اصلی خارج شده‌اند بطوریکه بتوانیم آنها را تحت کنترل درآوریم. ما همچنین ارزیابی می‌کنیم تا بتوانیم دستیابی به اهداف کیفیتی را برآورد کرده و تأثیرات پیشرفتهای فرآیندی و میتنی بر فنآوری را به محصولات و فرآیندها تخمین بزنیم.

ما پیش‌بینی می‌کنیم تا بتوانیم طرح‌ریزی کنیم. اندازه گیری بمنظور پیش‌بینی مستلزم کسب درک و فهم درمورد روابط بین پردازشها و محصولات و ساختن مدلهایی برای این روابط است، بطوری که ارزشهایی که برای برخی ویژگیها بدست می آوریم می توانند برای پیش‌بینی ویژگیهای دیگر بکار روند. ما این عمل را انجام می‌دهیم زیرا می‌خواهیم اهداف قابل دستیابی را برای هزینه‌ها، برنامه و کیفیت تعیین نمائیم – بطوریکه منابع مناسبی را بتوانیم بکار ببریم. اندازه‌های قابل پیش‌بینی همچنین پایه و اساس پیش‌بینی روندها هستند، بنابراین می‌توان تخمین‌هایی را برای هزینه‌ها، زمان و کیفیت براساس شواهد موجود انجام داد. طرح‌ریزی‌ها و تخمین‌هایی که براساس داده‌های تاریخی هستند به ما کمک می‌کنند تا خطرات را مورد تحلیل قرار داده و طراحی یا هزینه‌ها را با هم تنظیم کنیم.

ما اندازه‌گیری می‌کنیم تا بهبود و پیشرفت حاصل شود، هنگامی که اطلاعات کمیتی را جمع‌آوری می‌کنیم، به ما کمک می‌نمایند موانع، عوامل ریشه‌ای و نواقص را شناسایی نمائیم و فرصتهایی برای بهبود کیفیت محصول و عملیات پردازشی پیدا نمائیم.

نقل قول

متریک های نرم افزار به شما می گوید که چه موقع وقت خندیدن است و چه هنگام وقت گریستن.
تام گیبل

1.Park, R.E.

۴-۱ اندازه ها، متریک ها و معیارها

اگرچه واژه‌های «اندازه[1]، اندازه‌گیری[2] ومتریک[3]» اغلب به جای یکدیگر بکار گرفته شده‌اند، ولی ذکر اختلاف ناچیز بین آنها اهمیت دارد. بدلیل آنکه اندازه‌گیری و اندازه‌گیری کردن را می‌توان بعنوان اسم یا فعل بکار برد، تعریف این واژه‌ها می‌تواندگیج کننده باشد. در زمینه مهندسی نرم افزار، یک اندازه یک شاخص کمیتی از مقدار، میزان، ابعاد، ظرفیت یا اندازه یکی از ویژگیهای یک محصول یا فرآیند را فراهم می‌آورد. اندازه‌گیری کردن همان عمل تعیین یک اندازه است. فرهنگ واژگانی استاندارد[4] IEEE [IEE93][5] برای اصطلاحات مهندسی نرم‌افزار متریک را بصورت ذیل تعریف می‌کند :«یک اندازهٔ کمیتی از میزانی که یک سیستم، دستگاه یا فرآیند توسط آن یک ویژگی را کسب می‌کند».

نقل قول

نه هرآنچه شما می شمارید، شمردنی است و نه هرآنچه شمردنی است، می شمارید.
آلبرت انیشتین

هنگامی که یک امتیاز داده‌ای منفرد (مثل تعداد خطاهایی که در بررسی یک پیمانه منفرد مشخص نشده‌اند) جمع‌آوری می شود، یک اندازه‌گیری انجام شده است. اندازه‌گیری در نتیجه جمع‌آوری یک یا چند امتیاز دادهها انجام می‌شود. یک متریک نرم افزاری، اندازه‌های منفرد را به نحوی به یکدیگر مرتبط می‌کند.[6]

یک مهندس نرم‌افزار اندازه‌ها را جمع‌آوری کرده و متریک ها را تکمیل می‌کند بطوریکه علامتهایی شاخص بدست آیند. یک معیار شاخص[7]، یک متریک یا ترکیبی از متریک ها است که برای فرآیند نرم افزاری، یک پروژهٔ نرم‌افزار [RAG95][8] یا محصول برای ما بینش کافی را فراهم می‌آورند. یک معیار شاخص، همان بینشی را بوجود می آورد که به مدیر پروژه یا مهندسان نرم‌افزار امکان می دهد فرآیند یا پروژه را تنظیم کرده و امور را به نحو بهتری انجام دهند.

برای مثال، چهار تیم نرم‌افزاری بر روی یک پروژهٔ نرم‌افزاری عظیم کار می‌کنند. هر تیم باید بررسی‌هایی را روی طراحی انجام دهد، ولی اجازه دارد فقط آن نوع بازبینی را انتخاب کند که مورد استفاده قرار خواهد داد. به‌محض آزمون اندازه‌های متریک، خطاهایی به ازای هر نفر- ساعت که آن تیم کار کرده است پیدا می‌شود، و مدیر پروژه متوجه می‌شود که آن دو تیمی که از روشهای بازبینی رسمی‌تر استفاده کرده‌اند دارای نرخ خطای ۴۰ درصد کمتر از تیمهای دیگر هستند. به فرض اینکه تمام پارامترها یکسان باشند، این نتیجه گیری به مدیر پروژه نشان می‌دهدکه روشهای رسمی بازبینی . بازده زمانی

1.measure
2.mesurement
3.metrics
4.Glassary of Software Engineering
5.IEEE Standard
۶. این مفروضات و دیگر اندازه ها، نفر – ساعتی که صرف شده، برای هر بازبینی جمع خواهد شد.
7.indicator
8.Ragland, B.

بیشتری نسبت به روشهای غیر رسمی‌تر در پی دارند. او ممکن است تصمیم بگیرد که به همهٔ تیم‌ها پیشنهاد کند از روش رسمی‌تر استفاده کنند. این متریک به مدیر بینش می‌دهد. همین بینش به تصمیم گیری آگاهانه‌تر منتهی می‌شود.

۴-۲ متریک ها در حوزه پروژه و فرآیند

اندازه گیری در دنیای مهندسی امری پیش پا افتاده تلقی می‌گردد. ما مصرف برق، وزن، ابعادفیزیکی، دما، ولتاژ، سیگنال ـ به ـ نسبت صدا . فهرستی از مسائل بی‌شمار را اندازه می‌گیریم. متاسفانه، اندازه گیری در دنیای مهندسی نرم‌افزار رواج کمتری یافته است. ما در مورد چیزی که باید اندازه بگیریم و ارزیابی اندازه‌های جمع‌آوری شده مشکل داریم.

متریک ها باید بخوبی جمع‌آوری شوند که فرآیند و نشانگرهای محصول را بتوان تعیین نمود. نشانگرهای فرآیندی[1] به یک سازمان مهندسی نرم‌افزار امکان می‌دهد تا در مورد کارآیی یک روند موجود بینش بدست آورند. آنها به مدیران و مهندسان امکان می‌دهند تا ارزیابی کنند چه چیزی مؤثر و چه چیزی غیر مؤثر است. متریک پردازشی در سرتاسر پروژه‌ها و طی زمان‌های طولانی جمع‌آوری می‌شوند. هدف آنها ایجاد علامتهایی است که به بهبود طولانی مدت درفرآیند نرم‌افزاری منجر گردد.

علامتهای پروژه[2] به مدیر پروژهٔ نرم‌افزاری امکان می‌دهد تا (۱) وضعیت یک پروژهٔ در حال اجرا را ارزیابی کند، (۲) خطرات بالقوه را پیدا کند، (۳) حیطه‌های مشکل آفرین را قبل از اینکه به بحران تبدیل شوند چاره کند، (۴) جریان کار یا وظائف را تنظیم کند، و (۵) توانایی تیم‌های پروژه را برای کنترل کیفیت محصولات نرم‌افزاری مورد ارزیابی قرار دهد.

در برخی موارد، از همان متریک های نرم‌افزاری می‌توان برای تعیین پروژه و سپس علامتهای پردازشی استفاده نمود. در واقع، اندازه‌هایی که توسط یک تیم پروژه جمع‌آوری شده‌اند و برای استفاده در پروژه به متریک ها تبدیل شده‌اند می‌توانند به افرادی که مسئولیت پیشرفت فرآیند نرم‌افزاری را بر عهده دارند منتقل گردند. به همین دلیل، بسیاری از همان متریک ها در دامنهٔ پروژه و فرآیند بکار می‌روند.

۴-۲-۱ متریک های فرآیند و بهبود فرآیند نرم افزار

تنها راه منطقی برای پیشبرد هرگونه فرآیند، اندازه‌گیری ویژگیهای خاص آن فرآیند، توسعهٔ مجموعه‌ای از متریک های معنی دار، و سپس استفاده از این متریک ها برای تهیهٔ علامتها و شاخص‌هایی است که باعث ایجاد یک راهبرد برای پیشرفت شوند. ولی قبل از اینکه در مورد این متریک های و تأثیر

ارجاع به وب

یک کتاب راهنمای جامع برای متریک های نرم افزار قابل پیاده کردن از آدرس زیر می باشد :
www.irr.nasa.gov/SWG/resources/NASA-GB-001-94.pdf

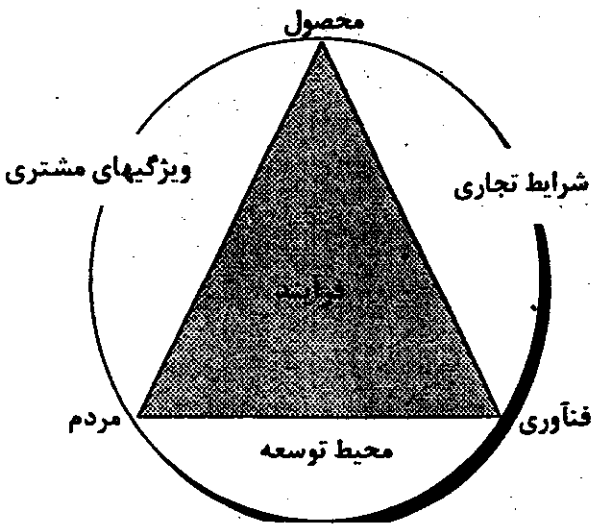
1.Proccess indicator

2.Project indicator

آنها بر پیشرفت فرآیند نرم‌افزاری بحث کنیم، لازم است ذکر نمائیم که فرآیند تنها یکی از چندین: عامل قابل کنترل در پیشبرد و بهبود کیفیت نرم‌افزار و عملکرد سازمانی؛ محسوب می‌گردد.[PAU94][1]

با اشاره به شکل ۴-۱ مشاهده می‌کنیم که فرآیند در وسط مثلثی قرار می‌گیرد که آن سه عامل با تأثیر شگرف بر کیفیت نرم‌افزاری و عملکرد سازمانی را به یکدیگر متصل می‌سازد. مهارت و انگیزه افراد بعنوان تنها عامل تأثیرگذار برکیفیت و عملکرد ذکر شده است.[BOE81][2] پیچیده بودن محصول می‌تواند تأثیری اساسی بر عملکرد تیم و کیفیت داشته باشد. آن فنآوری (یعنی روشهای مهندسی نرم‌افزار)که در سرتاسر مراحل پردازشی مشاهده می شوند نیز نوعی تأثیر دارند. علاوه بر اینها، مثلث فرآیند در میان یک دایره شرایط محیطی است که شامل: محیط توسعه (یعنی ابزارهای CASE)، شرایط تجاری (یعنی ضرب‌الاجل‌ها، قوانین تجاری) و ویژگیهای مشتری (یعنی تسهیل در ارتباط) می‌شود.

مهارت و انگیزه کسانی که مشغول کار می باشند، مهمترین فاکتور و عامل موثر در کیفیت نرم افزار محسوب می شود.



شکل ۴-۱ تعیین کیفیت نرم افزار و مؤثر بودن سازمانی (منطبق بر [PAU94])

ما کارآیی یک فرآیند نرم‌افزاری را بطور غیر مستقیم اندازه‌گیری می‌کنیم. بدین معنا که ، ما براساس نتایجی که می‌توان از فرآیند استنتاج نمود مجموعه‌ای از متریک ها را بدست می‌آوریم. این نتایج عبارتند از اندازه خطاهایی که قبل از تحویل نرم‌افزار مشخص نبودند، معایبی که به کاربران نهایی منتقل شده و آنها را گزارش کرده‌اند، محصولات کاری تحویل داده شده (بهره وری)، میزان تلاش انسان که صرف آن شده است، زمانی که صرف آن شده است، انطباق برنامه و دیگر اندازه‌ها. ما همچنین متریک های پردازشی را از طریق اندازه‌گیری ویژگیهای وظایف مهندسی نرم‌افزار بدست می‌آوریم. برای مثال، ما

چگونه می توانیم موثر بودن یک فرآیند نرم افزاری را اندازه گیری کنیم؟

1.Paulish, D.and A.

2.Boehm, B.

ممکنست تلاش و زمان صرف شده برای اجرای فعالیتهای چتری (پوششی) و فعالیتهای مهندسی نرم‌افزار را که در فصل ۲ توصیف نمودیم، اندازه‌گیری کنیم.

گریدی [GRA92][1] استدلال کرده است که کاربردهای «عمومی و خصوصی» برای انواع مختلف داده های پردازشی وجود دارند. بدلیل آنکه طبیعی به نظر می‌رسد که مهندسان نرم‌افزار نسبت به استفاده از متریک هایی که براساس یافته‌های فردی جمع‌آوری شده‌اند حساسیت داشته باشند. این داده‌ها باید کاملاً شخصی بوده و بعنوان علامتی برای خود آن فرد مورد استفاده قرار گیرند. مثالهای **متریک های خصوصی**[2] عبارتند از: نرخ معایب (توسط فرد)، نرخ معایب (توسط پیمانه)، و خطاهایی که طی پیشرفت یافت می شوند.

فلسفه «داده‌های پردازشی خصوصی» به خوبی با رهیافت **فرآیند نرم‌افزار شخصی**[3] که توسط هامفری [HUM95] ارائه گردیده، منطبق می‌شود. هامفری این رهیافت را بصورت ذیل توصیف کرده است:

فرآیند نرم‌افزار شخصی (PSP) یک مجموعه ساخت‌یافته از توصیف‌های پردازشی، اندازه‌گیری‌ها و روشهایی که می‌توانند به مهندسان در پیشبرد عملکرد شخصی‌شان کمک نمایند. این مجموعه، فرم‌ها دست نوشته‌ها و استانداردهایی را که به آنها کمک می‌کنند تا کارشان را برآورده و طرح‌ریزی نمایند، فراهم می‌آورد. این مجموعه به آنها نشان می‌دهد که چگونه فرآیندها را تعریف کرده و کیفیت و بهره وری آنها را اندازه‌گیری نمایند. یکی از اصول اساسی PSP آنست که همه افراد با یکدیگر فرق دارند و روشی که برای یک مهندس مؤثر و مفید است ممکن است برای مهندسی دیگر مناسب نباشد. بنابراین PSP به مهندسان کمک کند تا کار خودشان را اندازه گیری و ارزیابی کنند بطوریکه بتوانند روشهایی را که برایشان بهترین روشها بشمار می‌آیند، پیدا کنند.

هامفری می‌داند که پیشبرد فرآیند نرم‌افزاری می‌تواند و باید در سطح فردی آغاز گردد. داده‌های پردازشی خصوصی، هنگامی که یک مهندس نرم‌افزار قصد بهبود و پیشرفت دارد، می‌توانند بعنوان یک محرک پراهمیت عمل نمایند.

برخی از متریک های پردازشی برای تیم پروژه نرم افزاری، خصوصی ولی برای همه اعضای تیم عمومی هستند. مثالهای آن عبارتند از: معایبی که برای عملکردهای نرم‌افزار عمده (که توسط چندین نفر تکمیل کرده‌اند) گزارش می‌شوند، خطاهایی که طی بررسی تکنیکی رسمی یافت می‌شوند، و خطوط کد

متریک های عمومی یک سازمان را قادر می سازند که تغییرات راهبردی را که موجب پیشرفت فرآیند نرم افزار می شوند، اعمال کنند و نیز تغییرات تکنیکی و فنی را طی یک پروژه نرم افزاری ایجاد نمایند.

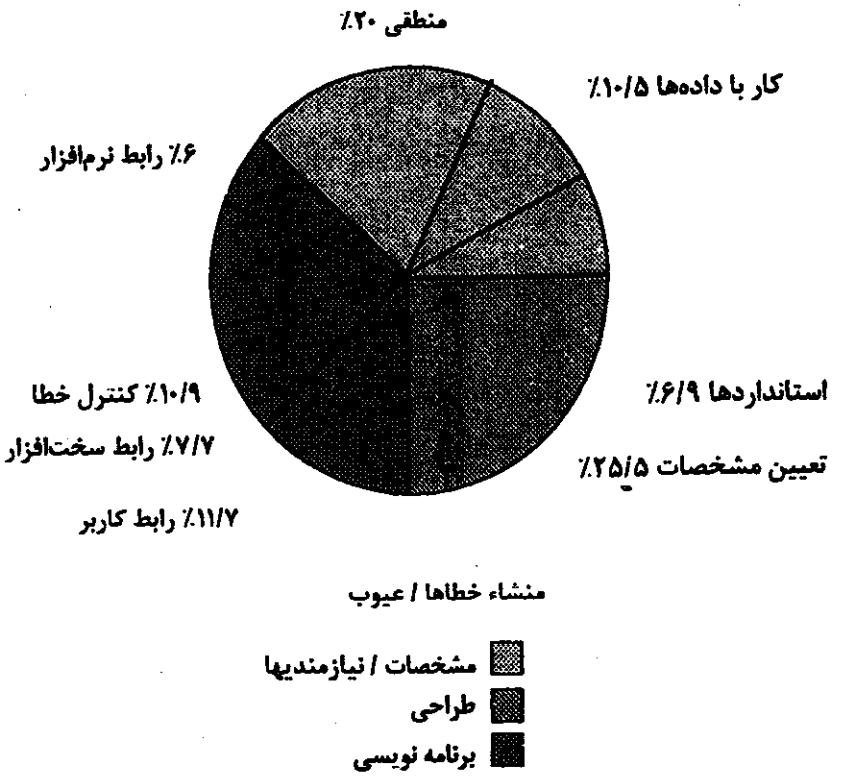
1.Grady, R.G.

2.private metrics

3.personal software process

برنامه یا امتیازات عملکردی در هر پیمانه وفانکشن[1] این داده‌ها بمنظور کشف علامتهایی که می‌توانند عملکرد تیم را بهبود بخشد توسط تیم بررسی می‌شوند.

متریک های عمومی[2] معمولاً اطلاعاتی را که در اصل برای اشخاص و تیم ها خصوصی محسوب می‌شده‌اند شبیه‌سازی می‌کنند. نرخ‌های عیب سطح پروژه‌نیروی کاری صرف شده، تقویم زمانی و داده‌های مربوطه، در تلاش برای آشکار نمودنِ علامتهایی که می توانند عملکرد پردازشی سازمان را بهبود بخشند جمع‌آوری شده و مورد ارزیابی قرار می‌گیرند.



شکل ۴-۲ دلایل عیوب و منشاء آنها برای چهار پروژه نرم‌افزاری [GRA 94]

۱. برای توضیح بیشتر در خصوص متریکهای تعداد خطوط برنامه (LOC) و امتیاز کارکردی (FP) بخشهای ۴-۳-۱ و ۴-۳-۲ را مطالعه نمایید.

2.public metrics

متریک های فرآیند نرم افزار، هنگامی که یک سازمان برای بهبود سطح کلی تکمیل فرآیند کار می‌کند، می‌تواند فواید عمده‌ای فراهم آورد. درهر صورت، مثل همه متریک ها، اینها نیز می‌توانند اشتباه بکار گرفته شوند و مشکلات بیشتری را بجای اینکه حل کنند ایجاد نمایند. گریدی[1] [GRA92] یک «اتیکت یا برچسب های متریک های نرم‌افزاری» را پیشنهاد کرده است که برای مدیران، هنگامی که یک برنامه متریک های پردازشی را ایجاد می‌کنند مناسب است:

- در موقع تفسیر داده‌های متریک ها از شعور و حساسیت سازمانی استفاده کنید.
- برای افراد و تیم هایی که برای جمع‌آوری اندازه‌ها و متریک ها کار کرده‌اند بازخور منظم تهیه کنید.
- برای ارزیابی افراد هرگز از متریک ها استفاده نکنید.
- با افراد و تیم‌ها برای تعیین اهداف واضح و متریک هایی که برای دستیابی به هدفها بکار می‌روند کار کنید.
- هرگز از متریک ها برای ترسانیدن و تهدید افراد و تیم‌ها استفاده نکنید.
- داده‌های متریک را که یک حیطه مشکل‌آفرین را نشان می‌دهند، نباید «منفی» درنظر گیرید. این داده‌ها صرفاً یک علامت برای پیشرفت فرآیند هستند.

پرسش

چه رهنمودهایی هنگام جمع کردن متریک های نرم افزار باید بکار رود؟

همانطور که یک سازمان با جمع‌آوری و کاربرد متریک های پردازشی مطلوب تر خواهد شد، اشتقاق معیارهای ساده نیز راهی به سوی یک رهیافت جدی‌تر می‌گشاید که آنرا پیشرفت فرآیند نرم‌افزار آماری (SSPI)[2] می‌نامیم. در اصل، SSPI از تحلیل خرابی نرم‌افزار برای جمع‌آوری اطلاعاتی درباره همه خطاها و معایبی[3] استفاده می‌کند که بعنوان یک برنامه کاربردی، سیستم، یا محصول توسعه داده شده و مورد استفاده قرار گرفته‌اند. تحلیل خربی به روش ذیل صورت می‌گیرد:

۱- همه خطاها و معایب بسته به منشاءشان طبقه بندی می‌شوند (یعنی، نقص در مشخصات، نقص در منطق، عدم انطباق با استانداردها).

۲- هزینه تصحیح هر خطا و عیب ثبت می‌شود.

۳- تعداد خطاها و معایب در هر طبقه شمرده می‌شود و آنها به ترتیب از بالا به پائین رده بندی می‌شوند.

ارجاع به وب

هر بهبود آماری فرآیند نرم افزار (SSPI) و دیگر اطلاعات مرتبط کیفی در انجمن آمریکایی کیفیت یافت می شود:
www.asq.org

---

1. Grady, R.
2. statistical software process improvement(SSPI)
۳. همانطور که در فصل ۸ توضیح خواهیم داد، یک خطا مشکل یک محصول است که پیش از تحویل به کاربر نهایی تشخیص داده می شود و یک نقص یا عیب، خطایی است که پس از تحویل به کاربر نهایی شناخته شود.

توصیه

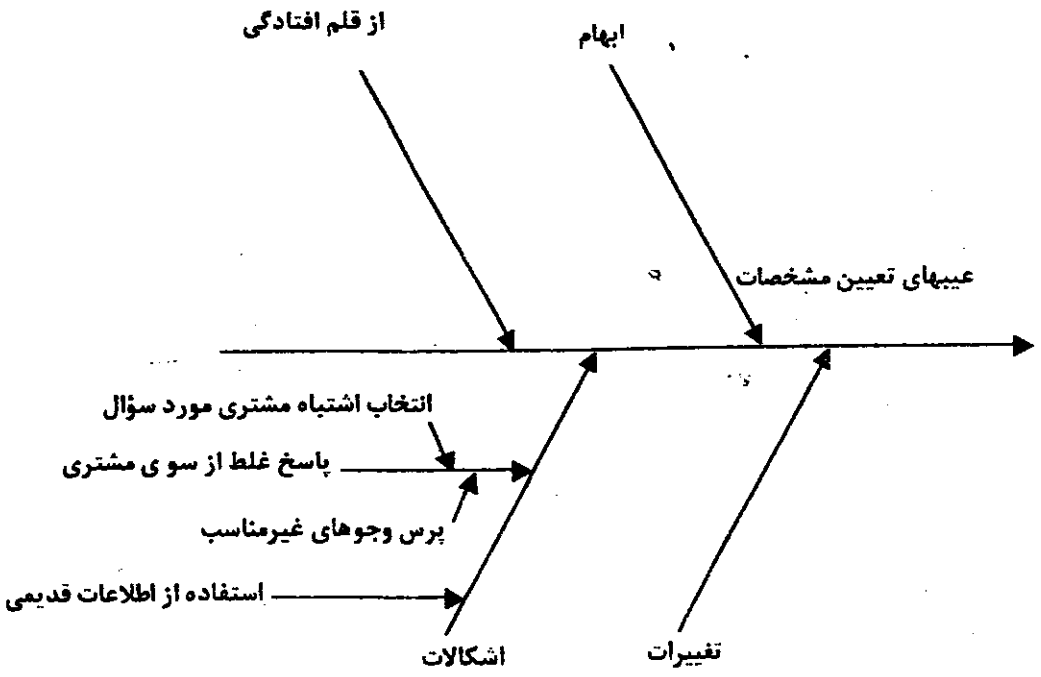
شما نمی توانید رهیافت مهندسی نرم افزار را بهبود بخشید، مادام که نقاط قوت و ضعف خود را بدانید. فنون بهبود آماری فرآیند نرم افزار (SSPI) را برای این منظور به کار برید.

۴- هزینه کلی خطاها و معایب در هر طبقه محاسبه می‌شود.

۵- داده‌های حاصله بمنظور پیداکردن طبقه‌هایی که بالاترین هزینه را برای سازمان بوجود می‌آورند مورد تحلیل قرار می‌گیرند.

۶- طرحهایی بوجود می‌آیند تا فرآیند، با هدف حذف آن طبقه از خطاها و معایبی که پرهزینه‌تر هستند اصلاح شود.

با پیروی از مراحل ۱ و ۲ بالا، یک توزیع عیب را می‌توان تکمیل نمود. در نمودار کیکی (که در شکل ۴-۲ آورده) [GRA94] هر علت و عیب و منشاء آنها نشان داده شده است. گریدی برای کمک به تشخیص داده‌هایی که در نمودار [GRA92] فرکانس و درصد عیوب نشان داده شده‌اند، توسعه و تکمیل یک نمودار استخوان ماهی[1] را پیشنهاد کرده است. با اشاره به شکل ۴-۳، استخوان ستون فقرات دیاگرام (خط مرکزی) عامل کیفیت را که تحت بررسی قرار دارد نشان می‌دهد. هر یک از استخوانها (خطهای مورب) که به ستون فقرات وصل شده‌اند دلایل بالقوه برای مشکل کیفیت را نشان می‌دهند. توضیح استخوانهای ماهی را بعداً به استخوانهای اصلی نمودار می‌افزائیم تا دلیل ذکر شده را توسعه دهیم. توسعه را فقط برای علت عدم صحت و نادرستی در شکل ۴-۳ نشان داده‌ایم.



شکل ۴-۳ یک نمودار استخوان ماهی (مطابق با [GRA 92])

1.fishbone diagram

مجموعه متریک های پردازشی، محرکی برای ایجاد یک نمودار استخوان ماهی است. برای بدست آوردن معیارهایی که به یک سازمان نرم افزاری، امکان می‌دهند بمنظور کاستن از فرکانس خطاها و معایب، فرآیند را اصلاح کند، یک نمودار استخوان ماهیِ کامل را می‌توان مورد تحلیل قرار داد.

۴-۲-۲ متریک‌های پروژه

متریک های فرآیند نرم‌افزار برای مقاصد راهبردی بکار می‌روند. اندازه‌های پروژه نرم‌افزار تاکتیکی هستند. یعنی، متریک های پروژه و معیارهای مشتق شده از آنها برای تطبیق جریان کار پروژه و فعالیتهای تکنیکی، توسط یک مدیر پروژه و تیم نرم‌افزار مورد استفاده قرار می‌گیرند.

ارجاع به فصلها

تکنیکهای تخمین و برآورد پروژه در فصل ۵ تشریح شده اند.

اولین برنامه کاربردی متریک های پروژه برای اکثر پروژه‌های نرم‌افزاری، طی برآورد بوجود می‌آید. متریک های جمع‌آوری شده از پروژه‌های گذشته بعنوان یک پایه و اساس بکار می‌روند که تخمین‌ها و برآوردهای زمان و تلاش طبق آن برای کار نرم‌افزاری اخیر بکار می‌روند. با پیشرفت پروژه، اندازه‌های زمان و تلاش به کاررفته با برآوردهای اصلی مقایسه می‌شوند. مدیر پژوژه این داده‌ها را بمنظور نظارت و کنترل بر پیشرفت مورد استفاده قرار می‌دهد.

هنگامی که کار تکنیکی آغاز می‌شود متریک های دیگر پروژه اهمیت می‌یابند. نرخهای تولیدی که بصورت تعداد صفحات اسناد، ساعتهای بررسی، امتیازات عملکردی و خطوط منبع تحویلی هستند اندازه‌گیری می‌شوند. بعلاوه، خطاهایی که طی وظیفه مهندسی نرم‌افزار کشف نشده‌اند، ردیابی می‌شوند.

هنگامی نرم‌افزار از مشخصات به طراحی تکامل می‌یابد، متریک های تکنیکی بمنظور سنجش کیفیت طراحی و فراهم آوردن معیارهایی که رهیافت اتخاذ شده برای برنامه نویسی و آزمون آن را ارزیابی نمایند، جمع‌آوری می‌شوند.

چگونه می توان متریک هارا طی یک پروژه، برای خود آن پروژه بکار برد؟

هدف از متریک های پروژه دو جنبه دارد. اول اینکه، متریک ها برای به حداقل رساندن زمان برنامه پیشرفت از طریق انجام تطبیق‌های لازم برای پرهیز از تأخیرها و کم کردن خطرات و مشکلات بالقوه بکار می‌روند. دوم اینکه، متریک های پروژه بمنظور سنجش کیفیت محصول بصورت مستمر و اصلاح رهیافت فنی برای بهبود کیفیت درصورت لزوم، مورد استفاده قرار می‌گیرند.

هرگاه کیفیت بهبود یابد، عیب‌ها به حداقل می‌رسند و هرگاه تعداد عیبها کم شد، مقدار کار مجددکه در طول پروژه لازم می‌شود کاهش می‌یابد. این امر به کاهش هزینه‌های پروژه منجر می‌گردد.

یکی دیگر از مدلهای متریک های پروژه نرم‌افزاری [HET93] [1] می‌گوید که هر پروژه باید موارد ذیل را اندازه‌گیری نماید:

- ورودی ها[2] - اندازه منابع (یعنی افراد، محیط) لازم برای انجام کار.

1. Hetzel, W.
2. Inputs

- **خروجی‌ها**[1]- اندازه‌گیریهای محصولات کاری یا قابل حمل که طی فرآیند مهندسی نرم‌افزار تولید می‌شوند.
- **نتایج**[2]- اندازه‌هایی که مؤثر بودن کالاهای قابل حمل و تحویل را بیان می‌کنند.

در عالم واقعیت، این مدل را می‌توان برای فرآیند و پروژه بکار برد. درزمینه پروژه، این مدل را می‌توان هنگامی که هر فعالیت چهارچوب دار انجام می‌شود بکار گرفت. بنابراین محصولات یک فعالیت بصورت مواد اولیه فعالیت بعدی درمی‌آیند. نتایج حاصل از متریک ها را می‌توان برای تهیه یک معیار برای مفید بودن محصولات کاری در موقعی که از یک فعالیت چهارچوب‌دار به فعالیت دیگر جریان پیدا می‌کنند، بکار برد.

۴-۳ اندازه گیری نرم افزار

پرسش

چه تفاوتی میان اندازه های مستقیم و غیر مستقیم وجود دارد؟

اندازه گیری در دنیای مادی (فیزیکی) را می‌توان به دو طریق اندازه‌های مستقیم (مثل طول یک پیچ) و اندازه‌های غیرمستقیم (مثل کیفیت پیچ‌های تولیدی، که بوسیله شمارش مرجوعی ها، اندازه‌گیری می‌شود) طبقه بندی نمود. متریک های نرم‌افزار را هم به همین نحو می‌توان طبقه‌بندی نمود.

اندازه‌های مستقیم[3] فرآیند مهندسی نرم‌افزار شامل هزینه و تلاش بکار رفته می‌شوند. اندازه‌های مستقیم محصول عبارتند از: خطوط کُد تولیدی، سرعت اجرا، اندازه حافظه و معایبی که در یک واحد زمانی گزارش شده‌اند. *اندازه‌های غیرمستقیم*[4] محصول عبارتند از: عملکرد، کیفیت، پیچیدگی، کارآیی، قابلیت اطمینان، قابلیت تعمیر و نگهداری و بسیاری از «توانایی‌های» دیگر که در فصل ۱۹ ذکر خواهیم کرد.

توصیه

یک کار نرم افزاری تحت تاثیر عوامل و فاکتورهای متعددی قرار دارد، بنابرین متریک‌ها را وسیله‌ای برای مقایسه افراد یا تیمها قرار ندهید.

ما قبلاً متریک های نرم‌افزار را بصورت: فرآیند، پروژه و متریک های محصول تقسیم‌بندی نمودیم. نیز ذکر نمودیم که آن متریک محصول که برای یک فرد خصوصی بشمار می‌آیند اغلب ترکیب می شوند تا متریک های پروژه‌ای که برای یک تیم نرم‌افزاری عمومی بشمار می‌آیند، را بوجود آورند. سپس متریک های پروژه برای ایجاد متریک های فرآیندی که برای یک سازمان نرم‌افزاری عمومی هستند، ادغام می‌گردند. ولی یک سازمان چگونه متریک هایی را که از پروژه‌ها یا افراد مختلف می‌گیرد ترکیب می‌کند؟

برای توضیح این مطلب، یک مثال ساده را بررسی می‌کنیم. افراد در دو تیم پروژه‌ای مختلف همه خطاهایی را که طی پروژه نرم‌افزاری می‌یابند، ثبت و طبقه‌بندی می‌کنند. سپس اندازه‌های افراد برای تکمیل اندازه‌های تیمی با هم ترکیب می‌شوند. تیم A طی فرآیند نرم‌افزاری قبل از آزادسازی (تکمیل محصول و ارائه به بازار) ۳۴۲ خطا یافت. تیم B، ۱۸۴ خطا یافت. درصورتیکه همه چیزهای دیگر مساوی

1.Outputs
2.Results
3.Directmeasures
4.Indirectmeasures

باشند، کدام تیم در پیدا کردن خطاها موفق تر است؟ بدلیل آنکه ما اندازه و میزان پیچیدگی پروژه‌ها را نمی‌دانیم، نمی‌توانیم به این پرسش پاسخ دهیم. درهر صورت اگر اندازه‌ها را بصورت نرمال درآوریم، امکان ایجاد متریک های نرم‌افزاری که مقایسه با میانگین‌های سازمانی وسیع‌تر را میسر می‌سازد، فراهم می‌آید.

پرسش

برای کار با متریک های مبتنی بر اندازه (سایز) چه داده هایی را باید جمع آوری نماییم؟

۴-۳-۱ متریک های مبتنی بر اندازه

متریک های نرم‌افزاری مبتنی بر اندازه، از طریق نرمال کردن کیفیت و یا اندازه‌های بازدهی بوسیله بررسی سایز[1] نرم‌افزار تولید شده مشتق‌گیری می‌شوند. اگر یک سازمان نرم‌افزاری پرونده‌های ساده را نگهداری کند، یک جدول از اندازه‌های مبتنی بر اندازه‌گیری، نظیر جدول شکل ۴-۴ را می‌تواند ترسیم نماید.

این جدول هریک از پروژه‌های توسعه نرم‌افزاری را که طی چند سال گذشته تکمیل شده‌اند و اندازه‌گیری همسان برای آن پروژه را فهرست بندی می‌کند. با اشاره به مدخل جدول (شکل ۴-۴)

برای پروژه آلفا (الف) می‌بینیم که ۱۲/۱۰۰ خط برنامه (LOC)[2] با تلاش ۲۴ نفر در ماه با هزینه ۱۶۸,۰۰۰ دلار تکمیل گردید. لازم به ذکر است که تلاش و هزینه ثبت شده در جدول، همه فعالیتهای مهندسی نرم‌افزار را نشان می‌دهد. اطلاعات بیشتر برای پروژه آلفا نشان می‌دهد که ۳۶۵ صفحه سند تکمیل شده است، ۱۳۴ خطا قبل از اینکه نرم‌افزار تحویل داده شود و ۲۹ عیب پس از تحویل به مشتری در اولین سال راه‌اندازی دیده شده سه نفر بر روی توسعه نرم‌افزار برای پروژه آلفا (الف) کار می‌کردند.

1.size

2.Line Of Code

پروژه	تعداد خطوط برنامه	نیروی به کار رفته	هزینه(دلار)	تعداد صفحات مستند سازی	تعداد خطها	تعداد عیبها	تعداد افراد دخیل
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
•	•	•	•	•	•		
•	•	•	•	•	•		
•	•	•	•	•	•		

شکل ۴ـ۴ متریکهای مبتنی بر سایز

قالب جمع شده متریک ها

ما برای توسعه متریک هایی که بتوان با متریک های مشابه از دیگر پروژه‌ها آنرا شبیه سازی نمود، خطوط کُد را بعنوان ارزش نرمال سازی انتخاب کردیم. از داده‌های مقدماتی و اولیه که در جدول وجود دارند، مجموعه‌ای از متریک های ساده مبتنی بر اندازه را می‌توان برای هر پروژه تکمیل نمود:

- خطاها در KLOC (هزار خط برنامه).
- تعداد عیب ها[1] در هر هزار خط برنامه (KLOC).
- هزینه هر خط برنامه (LOC) (بر حسب دلار یا واحد پول).
- تعداد صفحات مستندات به ازای هر هزار خط برنامه (KLOC).

متریک های مبتنی بر اندازه بطور گسترده ای مورد استفاده دارند. اما بحث بر سر اعتبار و کارآمدی آنها ادامه دارد.

علاوه بر اینها، متریک های دیگری را هم می‌توان محاسبه نمود:

- تعداد خطاها به ازای هر نفر - ماه.
- تعداد خطوط برنامه (LOC) به ازای هر نفر - ماه.
- هزینه هر صفحه از مستندات (بر حسب دلار یا واحد پول).

متریک های مبتنی بر اندازه برای اندازه زدن فرآیند توسعه نرم افزار مورد اقبال همگان نمی باشند.[JON86] مباحثه و جدل هایی در خصوص استفاده از تعداد خطوط برنامه وجود دارد. استدلال طرفداران استفاده از اینگونه متریک ها (تعداد خطوط) آن است که حاصل فرآیند نرم

۱. یک عیب و نقص‌هنگامی روی می دهدکه فعالیتهای تضمین کیفیت (برای مثال بازبینی های فنی رسمی) طی فرآیند نرم افزار، خطا را نیابند و بالطبع پوشش ندهند.

افزاری همان کد برنامه است و شمارش آن ساده و روشن می باشد و از اینرو بسیاری از مدل های تخمینی نرم افزار از این متریک بهره می برند. در مقابل مخالفان این امر معتقدند که تعداد خطوط به زبان برنامه سازی وابسته است و برنامه ای با طراحی خوب تعداد خطوط کمتری خواهد داشت. خصوصاً" اگر از زبانهای غیر رویه ای استفاده شود تخمین مشکل خواهد بود و به جزئیاتی نیاز است که دسترسی و استفاده از آنها و مدل سازی آن دشوار و طاقت فرسا می نماید. (ضمن آنکه برنامه ریز باید به نحوی ارتباطی بین فعالیت های تحلیل و طراحی و تعداد خطوط برنامه برقرار سازد که این تخمین و برآورد مشکل خواهد بود).

۴-۳-۲ متریک های مبتنی بر کارکرد (کارکرد ـ محور)

متریک های نرم‌افزاری مبتنی بر کارکرد از یک معیار کارکردی که بوسیله برنامه نرم‌افزاری بعنوان یک ارزش نرمال سازی تحویل می‌گردد، استفاده می‌نمایند. از آنجا که عملکرد را نمی‌توان مستقیماً اندازه‌گیری نمود، باید بطور غیر مستقیم با استفاده از اندازه‌های مستقیم دیگر اندازه‌گیری شود. متریک های مبتنی بر کارکرد برای اولین بار توسط آلبرشت [ALB79][1] پیشنهاد شدند، که وی یک اندازه را که امتیازات عملکردی[2] نامیده می‌شد پیشنهاد کرد. امتیازات عملکردی با استفاده از یک رابطه تجربی براساس اندازه‌های قابل شمارش (مستقیم) دامنه اطلاعات نرم‌افزاری و ارزیابی پیچیدگی نرم‌افزاری به دست می‌آیند. امتیازات عملکردی[3] [IFP94] بوسیله تکمیل جدول نشان داده شده در شکل ۴-۵ محاسبه می‌شوند. پنج ویژگی دامنه اطلاع رسانی تعیین شده‌اند و اعدادی در محل مناسب جدول آمده است. ارزشهای دامنه اطلاع‌رسانی رابه روش ذیل تعریف کرده‌ایم:[4]

تعداد ورودی‌های کاربر: هر ورودی کاربری که داده‌های محوری و مورد نیاز برنامه نرم‌افزاری را فراهم می‌سازد، شمرده می‌شود. داده‌های ورودی‌ها بایستی با سئوالایی که جداگانه پرسیده می‌شوند فرق داشته باشند.

تعداد خروجی‌های کاربر: هرگونه خروجی کاربر که اطلاعات مبتنی بر برنامه کاربردی نرم‌افزار را فراهم می‌آورد، شمرده می‌شود. در این متن خروجی به گزارشها، بررسی‌ها، پیام‌های خطا و غیره اشاره دارد. داده‌های اطلاعاتی تکی را در یک گزارش بطور جداگانه نمی‌شمارند.

ارجاع به وب

اطلاعـــات جامعی در خصـــوص امتیازات کارکردی در آدرس زیر قابل دستیابی است www.ifpug.org

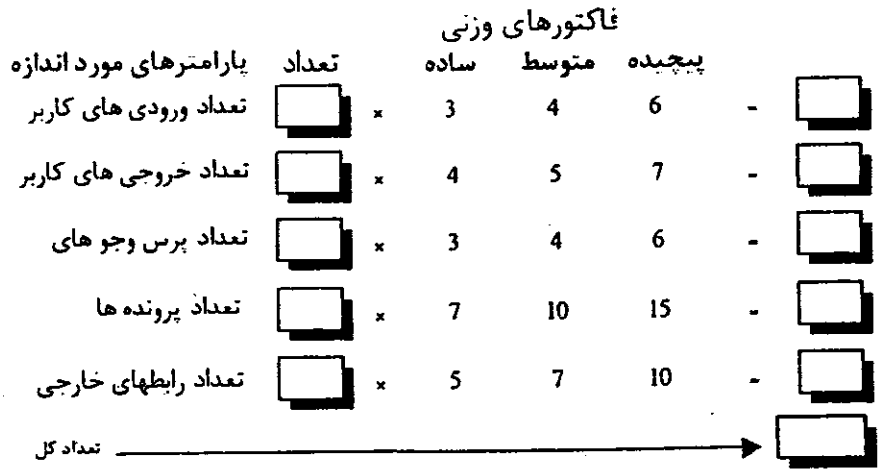
امتیازات کارکردی، از اندازه های مستقیم در حوزه اطلاعاتی بدست می آید.

1.Albrecht, A.J.

2.Function point

3.Function Point Counting

۴. در عمل، تعریف مقادیر حوزه اطلاعاتی و تاثیرات آنها قدری پیچیده است. خوانندگان علاقه مند برای جزئیات بیشتر به [IFP94] مراجعه کنند.



شکل ۴-۵ محاسبه امتیاز کارکرد (عملیاتی)

تعداد درخواست های کاربر: یک درخواست را بعنوان یک ورودی مرتبط تعریف کرده‌اند که باعث بوجود آوردن چندین پاسخ نرم‌افزاری فوری به شکل یک خروجی مرتبط می‌شود. هر درخواست مشخص شمرده می‌شود.

تعداد فایل‌ها: هرگونه فایل اصلی منطقی، یعنی گروه‌بندی منطقی داده هایی که ممکن است یکی از بخشهای یک پایگاه داده‌های بزرگ یا یک فایل جدا باشند شمارش می‌شود.

تعداد رابطه‌های خارجی: همه رابطهای قابل خوانده شدن توسط ماشین (یعنی فایلهای داده‌ها بر روی نوار یا دیسک) که برای انتقال اطلاعات به یک سیستم دیگر بکار می‌روند، شمارش می‌شوند.

هنگامی که داده‌های فوق جمع‌آوری شوند، یک مقدار پیچیدگی با هر شمارش همراه می‌شود. سازمانهایی که از روشهای امتیازات عملکردی استفاده می‌کنند، برای تعیین اینکه آیا یک مدخل ویژه ساده است یا متوسط یا پیچیده، معیارهایی را تکمیل می‌کنند. به هر حال، تعیین پیچیدگی تاحدی ذهنی است.

برای محاسبه امتیازات عملکردی (FP)، رابطه ذیل بکار می‌رود:

$$FP = \text{Count total} \times [0.65 + 0.01 \times \Sigma(F_i)] \qquad (۱-۴)$$

که تعداد کل (Count total)، مجموع همه مدخل‌های FP است که از شکل ۴-۵ بدست آمده‌اند.

F_i $(I = 1 - 14)$ « مقدار تنظیم پیچیدگی» بشمار می‌آیند که براساس پاسخ به سئوالهای ذیل هستند: [ART85][1]

1.Arthur, L.G.

۱- آیا سیستم به پشتیبان و احیاء و بازیابی قابل اطمینان نیاز دارد؟

۲- آیا ارتباطات داده‌ها موردنیاز است؟

۳- آیا عملکردهای پردازشی توزیع شده وجود دارند؟

۴- آیا عملکرد آنها ضروری و بحرانی است؟

۵- آیا سیستم در یک محیط عملیاتی کاربردی سنگین موجود اجرا شده است؟

۶- آیا سیستم به ورود داده‌های روی خط (on line) نیاز دارد؟

۷- آیا ورود داده‌های روی خط به تراکنش ورودی نیاز دارد تا در عملیاتها یا صفحه نمایشهای چندگانه ایجاد گردد؟

۸- آیا فایلهای اصلی بطور متصل و روی خط (on line) به هنگام سازی می شوند؟

۹- آیا ورودیها، خروجی‌ها، فایل‌ها یا درخواستها و پرس و جو ها پیچیده هستند؟

۱۰- آیا فرآیند درونی پیچیده است؟

۱۱- آیا کُد برنامه طوری طراحی شده که مورد استفاده مجدد قرار گیرد؟

۱۲- آیا تبدیل ها و نصب و پیاده سازی در طراحی منظور گردیده اند؟

۱۳- آیا سیستم برای نصب چندگانه در سازمان‌های مختلف طراحی شده است؟

۱۴- آیا برنامه نرم‌افزاری برای تسهیل تغییرات و سهولت استفاده توسط کاربر طراحی شده است؟

هریک از سئوالهای فوق با استفاده از یک مقیاس صفر تا ۵ پاسخ داده شده‌اند. ارزشهای ثابت که تحت سئوال هستند و عوامل موزون که برای تعدا دامنه‌های اطلاع‌رسانی بکار گرفته شده‌اند بطور تجربی تعیین شده‌اند.

هنگامی که امتیازات عملکردی محاسبه می‌شوند، آنها به روشی مشابه LOC بعنوان طریقی برای نرمال کردن اندازه ها جهت کیفیت، بهره وری نرم‌افزار و دیگر ویژگیها مورد استفاده قرار می‌گیرند:

- تعداد خطاها به ازای هر امتیاز کارکردی
- تعداد عیوب و نواقص به ازای هر امتیاز کارکردی
- تعداد صفحات مستندات به ازای هر امتیاز کارکردی
- تعداد امتیازات کارکردی به ازای هر نفر ماه

امتیازات کارکردی گسترش یافته، برای مهندسی ، زمان واقعی ، و کاربردهای مبتنی بر کنترل مورد استفاده دارد.

۴-۳-۳ متریک های توسعه یافته امتیاز کارکردی

اندازه امتیازات عملکردی در اصل برای کاربرد در برنامه های نرم‌افزاری سیستم‌های اطلاع رسانی تجاری، طراحی شده است. برای تطبیق این برنامه‌های کاربردی، ابعاد داده‌ها با میزان ابعاد عملکردی و رفتاری مورد تأکید قرار گرفت. به همین دلیل، اندازه امتیازات عملکردی برای بسیاری از مهندسان و

سیستم‌های درونی کافی نبود. چندین توسعه برای اندازهٔ امتیازات عملکردی اصلی برای اصلاح این وضعیت پیشنهاد شده‌اند.

یک امتیازات عملکردی توسعه یافته که امتیاز ویژگی[1] [JON91][2] نامیده می‌شود یک فوق مجموعه از اندازه امتیازات عملکردی است که می‌تواند برای سیستم و برنامه‌های کاربردی نرم‌افزار مهندسی بکار گرفته شود. اندازه امتیاز ویژگی آن برنامه‌های کاربردی را تطبیق می‌دهد که پیچیدگی الگوریتمی در آنها بالا است. زمان واقعی، کنترل فرآیند و برنامه های کاربردی نرم‌افزار جاسازی شده، دارای پیچیدگی الگوریتمی بالایی هستند و با امتیاز ویژگی قابل اصلاح هستند.

برای محاسبه امتیاز ویژگی، ارزشهای دامنه اطلاع رسانی را شمرده و همانطور که در بخش ۴-۳-۲ توضیح دادیم موزون می‌کنند. علاوه بر این متریک امتیاز ویژگی یک ویژگی نرم‌افزاری جدید – یعنی الگوریتمها[3] – را می‌شمارد. یک الگوریتم را بعنوان «یک مسئله محاسبه‌ای محدود مشتمل بریک برنامه کامپیوتر خاص» [JON91] [4] تعریف می‌کنند. بدست آوردن معکوس یک ماتریس، رمزگشایی یک رشته بیت، یا رفع و رجوع یک وقفه همگی نمونه‌هایی از الگوریتم هستند.

یکی دیگر از امتیازات عملکردی توسعه یافته برای سیستم‌های بلادرنگ و محصولات ساخته شده توسط کمپانی بوئینگ توسعه داده شده است. رهیافت بوئینگ مشتمل بر ابعاد داده‌های نرم‌افزاری با ابعاد کنترلی و عملیاتی برای فراهم آوردن یک اندازه مبتنی بر تابع است که برای آن برنامه‌های کاربردی که بر قابلیت‌های کنترل و عملکردی تأکید می‌ورزند قابل اصلاح است. ویژگیهای این سه ابعاد نرم‌افزاری که امتیازات عملکردی[5] سه بعدی [WHI95] [6] نامیده می‌شوند عبارتند از «شمارش شده – اندازه‌گیری شده و تبدیل شده» به اندازه ای که معیاری برای قابلیت عملکرد تحویل شده توسط نرم‌افزار تهیه می‌کند.[7]

ارجاع به وب

یک FAQ مفید برای امتیازات کارکردی (و امتیازات کارکردی توسعه یافته)در آدرس زیر قابل دستیابی است
http://ourwork.compuserve.com/homepages/softcomp/

1.feature points
2.Jones, C.
3.algorithms
4.Jones, C.
5.3D function point
6.Whitmire, S.A.

۷. باید توجه داشت که دیگر امتیازات کارکردی توسعه یافته نیز برای نرم افزارهای زمان واقعی [ALA97] وجود دارند. با این وجود کاربرد گسترده ای در صنعت ندارند.

جملات معنایی / گامهای پردازشی	۱-۵	۶-۱۰	+۱۱
۱ - ۱۰	پائین	پائین	متوسط
۱۱ - ۲۰	پائین	متوسط	بالا
+۲۱	متوسط	بالا	بالا

شکل ۴-۶ تعیین میزان پیچیدگی یک تبدیل برای از امتیازات عملیاتی سه بعدی [WHI95]

بعد داده‌ها[1] به همان طریقی که در ۴-۳-۲ توضیح دادیم محاسبه می‌شود. تعداد داده‌های باقیمانده و داده‌های بیرونی همراه با اندازه‌های پیچیدگی بکار می‌روند تا یک تعداد بعد داده‌ها[2] را حاصل کنند. بعد عملیاتی از طریق بررسی «تعداد عملیاتهای داخلی که برای تبدیل ورودی به داده‌های خروجی موردنیاز هستند» [WHI95][3] اندازه‌گیری می‌شود. به منظور محاسبه امتیازات عملکردی 3D یک «تغییر شکل» را بعنوان مجموعه‌ای از مراحل فرآیندی که توسط گروه‌ای از جملات معناشناسانه محدود می‌شوند، دانسته‌اند. بعد کنترل[4] از طریق شمردن تعداد جابجایی‌های بین وضعیت‌ها اندازه‌گیری می‌شود.[5]

یک وضعیت نوعی حالت رفتاری بیرونی حالت مشاهده را نشان می‌دهد و یک انتقال در نتیجه حادثه ای که سبب می‌شود نرم‌افزار یا سیستم حالت رفتاری اش را تغییر دهد بوجود می‌آید. برای مثال، یک تلفن بی‌سیم دارای نرم‌افزاری است که از عملکردهای شماره‌گیری خودکار[6] پشتیبانی می‌کند. برای وارد کردن وضعیت شماره‌گیری خودکار از یک وضعیت درحال استراحت[7]، کاربر یک **دکمه شماره‌گیری‌خودکار**[8] را در روی صفحه دکمه‌ها فشار می‌دهد. این امر سبب می‌گردد تا نمایشگر LCD یک کد را نشان دهد که بر مخاطب دلالت می‌کند. بمحض ورود کد و فشردن دکمه شماره‌گیری، نرم‌افزار تلفن بی‌سیم آنرا به

1.data dimension

2.Functional dimension

3.Whitmire, S.A.

4.control dimension

۵.توضیح مفصلی از بعد رفتاری، شامل وضعیتها و انتقال وضعیت را در فصل ۱۲ می توانید مشاهده نمایید.

6.auto-dial

7.westing state

8.dialing

وضعیت شماره‌گیری تبدیل می‌کند. هنگامی که امتیازات عملکردی 3D را محاسبه می‌کنیم، به جابجائی ها یک مقدار پیچیدگی نمی‌دهیم.

برای محاسبه امتیازات عملکردی 3D از رابطه ذیل استفاده می‌کنیم:

$$\text{index} = I + O + Q + F + E + T + R \qquad (۲-۴)$$

که I, O, Q, F, E, T و R مقادیر وزن دار پیچیدگی را برای عناصر فوق‌الذکر نشان می‌دهند یعنی: ورودیها، خروجی‌ها، درخواستها، ساختار داده‌های درونی، فایل‌های بیرونی، تبدیل و جابجایی. هر مقدار وزن دار پیچیدگی با استفاده از رابطه ذیل محاسبه می‌شود:

$$\text{complexiti weighted value} = N_{il}W_{il} + N_{ia}W_{ia} + N_{ih}W_{ih} \qquad (۲-۴)$$

که N_{il}، N_{ia} و N_{ih} تعداد وقوع عنصر i (یعنی مثل خروجی‌ها) را برای هر سطح پیچیدگی (کم، متوسط، زیاد) نشان می‌دهند و W_{il}، W_{ia} و W_{ih} وزنهای معادل آن هستند. محاسبه کلی برای امتیازات عملکردی 3D را در شکل ۴-۶ نشان داده‌ایم. باید ذکر کنیم که امتیازات عملکردی ، نقطه‌های ویژگی، و امتیازات عملکردی 3D یک چیز را نشان می‌دهند – یعنی «قابلیت عملیاتی یا «بکارگیری» که توسط نرم‌افزار حمل می‌شود. درواقع ، هر یک از این اندازه‌ها درصورتی که فقط بُعد داده‌ای یک برنامه کاربردی بررسی شود، همان ارزش را بوجود می‌آورند. برای سیستم‌های واقعی پیچیده‌تر، تعداد امتیازات عملکردی اغلب بین ۲۰ الی ۳۵ درصد بالاتر از تعدادی است که با استفاده از امتیازات عملکردی تعیین شده بود.

امتیازات عملکردی (و مدل توسعه یافته آن)، مثل اندازه LOC، مشاجره برانگیز و ضد و نقیض است. طرفداران آن اظهار می‌کنند که FP یک زبان برنامه نویسی، مستقل است که آن را برای برنامه‌های کاربردی با استفاده از زبانهای قراردادی و غیررویه‌ای بصورت ایده‌آل درمی‌آورد.

این زبان براساس داده هایی است که به احتمال زیاد در اوائل مراحل سیر تکامل یک پروژه شناخته می شوند و FB را جالب تر از «روش برآورد» می‌کند. مخالفان اظهار می‌کنند که این روش مستلزم «تردستی» است زیرا محاسبه بجای اینکه عینی باشد، بطور ذهنی است. داده‌ها که دامنهٔ اطلاع رسانی را مشخص می‌کنند، به سختی جمع‌آوری می‌شوند و FP هیچگونه معنی فیزیکی مستقیم ندارد.

۴-۴ تطبیق رهیافتهای مختلف متریک

رابطه بین خطوط کد و امتیازات عملکردی به زبان برنامه‌ریزی که برای اجراء نرم‌افزار و کیفیت طراحی بکار برده شده اند بستگی دارد. در چندین تحقیق تلاش شده است که اندازه‌های LOC و FP به یکدیگر مرتبط شوند. برای نقل قول جملات آلبرشت و گافنی[ALB83][1] : قضیه این کار در آن است که

1.Albrecht, A.J.

میزان عملکردی را که باید بوسیله برنامه کاربردی فراهم شود می‌توان از جزء به جزء نوشتن اجزاء اصلی[1] داده هایی که توسط آن استفاده یا فراهم می‌شوند، تخمین زد. علاوه بر این، تخمین عملکرد را باید برای توسعه دادن به مقدار LOC و به توسعه تلاش لازم مرتبط ساخت.

جدول زیر [JON98][2] تخمین های دقیق تعداد میانگین خطوط کد را که برای ساختن یک امتیازات عملکردی در زبانهای برنامه سازی متعدد موردنیاز هستند نشان می‌دهد:

پرسش

اگر تعداد خطوط برنامه را بدانیم آیا تخمین امتیازات کارکردی امکان پذیر است؟

زبان برنامه سازی	LOC/FP (میانگین تعداد خطوط به ازای هر امتیاز کارکردی)
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

با مروری بر داده‌های فوق درمی‌یابیم که تعداد خطوط C++ تقریباً ۱/۶ بیشتر نسبت به تعداد خطوط FORTRAN « امتیاز و قابلیت عملکردی » را ایجاد می‌کند. بعلاوه، تعداد خطوط مربوط به VISUAL BASIC، ۳ برابر بیش از تعداد خطوط یک زبان برنامه سازی سنتی امتیاز عملکردی خواهد داشت. داده‌های دقیق تر درمورد رابطه بین FP و LOC در [JON98] ارائه شده‌اند و می توانند برای محاسبه برنامه‌های موجود بکار روند تا اندازه FP هریک تعیین گردد.

توصیه

داده های پشتیبان را برای قضاوت انتخاب کنید، این امر بهتراز استفاده شیوه توضیح داده شده پیشین، خواهد بود.

اغلب اندازه‌های FP و LOC برای به دست آوردن متریک های بهره وری بکار می‌روند. این امر به یک مشاجره درمورد کاربرد چنین داده‌هایی منجر می‌گردد. آیا LOC یا نفر – ماه هر گروه را باید با داده‌های مشابه گروهی دیگر مقایسه نمود؟ آیا مدیران باید با استفاده از این متریک ها عملکرد افراد را ارزیابی کنند؟ پاسخ این سئوالها یک «نه» قاطعانه است. دلیل این پاسخ آن است که بسیاری از عوامل بر

۱. توجه به این امر مهم است که " اجزاء اصلی را به صورت اقلام بیان داشتن " به گونه های مختلفی ممکن است. برخی مهندسین نرم افزار، که در محیط های توسعه شئ گرا کار می کنند (بخش چهار) تعداد کلاسها یا اشیاء را به عنوان متریک اندازه حوزه انتخاب می کنند. سازمانهایی که نگهداری و پشتیبانی را عهده دار هستند، بزرگی پروژه را در تعداد درخواستهای تغییر (فصل ۹) میبینند. یک سازمان سیستمهای اطلاعاتی ممکن است تعداد فرآیندهای تجاری را مد نظر قرار دهد.

2.Jones, C.

بهره وری تأثیر می‌گذارند و این مقایسه مثل مقایسه «سیب با پرتقال» است که می‌تواند کاملاً اشتباه تفسیر شود.

ما امتیازات کارکردی و متریک های مبتنی بر LOC را معیار های نسبتاً صحیحی برای هزینه و نیروی کار بمنظور تکمیل نرم‌افزار می‌دانیم. در هر صورت بمنظور استفاده از LOC و FP در فنون تخمین (برآورد)، یک خط سیر اطلاع رسانی باید مشخص گردد.

۴-۵ متریک های کیفیت نرم افزار

هدف برتر مهندسی نرم‌افزار بوجود آوردن یک سیستم، برنامه کاربردی یا محصول با کیفیت بالا می باشد. برای رسیدن به این هدف، مهندسان نرم‌افزار باید روشهای مؤثری را که با ابزارهای مدرن منطبق هستند در زمینه یک فرآیند نرم افزاریِ کامل اتخاذ نمایند. بعلاوه، یک مهندس نرم‌افزار خوب بایستی ببیند که کیفیت بالا درک شده یا خیر.

کیفیت یک سیستم، برنامه کاربردی یا محصول، فقط با مفید بودن نیازمندیهای توصیف کننده مشکل، طراحی مربوطه به مدل سازی راه حل، مفید بودن کد برنامه قابل اجرا و آزمونهایی که خطاها را آشکار می‌کنند، به دست می آید. یک مهندس نرم‌افزار خوب از اندازه‌گیری برای ارزیابی کیفیت تحلیل و مدلهای طراحی، کد منبع و آزمونهایی که در موقع ساخت نرم‌افزار بوجود آمده‌اند استفاده می‌کند. برای تحقق این ارزیابی کیفیت واقعی، مهندس باید از اندازه‌های فنی (فصل ۱۹ و ۲۴) برای ارزیابی کیفیت به روشهای عینی و نه روشهای ذهنی استفاده نماید.

مدیر پروژه نیز بایستی کیفیت را در هنگام پیشرفت پروژه ارزیابی کند. متریک های خصوصی که توسط مهندسان نرم‌افزار جمع‌آوری شده‌اند برای فراهم آوردن نتایج سطح پروژه شبیه‌سازی می‌شوند. اگر چه اندازه‌گیریهای کیفیتی بسیاری را می‌توان جمع‌آوری نمود ولی اولین کار دشوار در

سطح پروژه اندازه‌گیری خطاها و معایب است. متریک هایی که از این اندازه‌ها مشتق شده‌اند، دلیلی بر مؤثر بودن فعالیتهای کنترل و اطمینان از کیفیت نرم‌افزار گروهی و فردی هستند.

متریک هایی چون تعداد خطاهای محصول کاری به ازای هر امتیاز کارکردی (نیازمندیها یا طراحی) ، خطاهای کشف شده به ازای هر ساعت بازبینی، و تعداد خطاهای کشف شده به ازای هر ساعت آزمون، در خصوص تاثیر و سودمندی فعالیتهایی که متریک های برایشان به کار رفته، بصیرت و آگاهی به دنبال خواهند داشت. داده های خطا برای به دست آوردن و محاسبه کارآیی رفع نقص (DRE) جهت هر فعالیت چارچوبی کاربرد خواهند داشت. DRE در بخش ۴-۳-۲ توضیح داده شده است.

یک منبع ممتاز اطلاعات مربوط به کیفیت نرم افزار و عناوین مرتبط (شامل متریکها) در آدرس زیر یافت می شود:

www.qualityworld.com

یک توصیف تفصیلی از فعالیتهای تضمین کیفیت نرم افزار در فصل ۸ ارائه گردیده است.

۴-۵-۱ نگاهی اجمالی به فاکتورهای موثر بر کیفیت

طی ۲۵ سال گذشته، کاوانو و مک کال [MCC78][1] مجموعه ای از عوامل کیفیتی را که اولین مرحله برای توسعهٔ متریک ها برای کیفیت نرم‌افزاری بشمار می‌آیند تعریف کرده‌اند. این عوامل نرم‌افزار را از سه جنبه بررسی می‌کنند: ۱- راه‌اندازی محصول ۲- بررسی محصول (تغییر در آن) و ۳- انتقال محصول (از محیطی به محیط دیگر) نویسندگان، در کارشان، رابطه بین این عوامل کیفیتی (از آنچه آنها چهارچوب کاری می‌نامند[2]) و دیگر جنبه‌های فرآیند مهندسی نرم‌افزار را توصیف می‌کنند:

اولاً، چهارچوب مکانیسمی برای مدیر پروژه فراهم می‌کند تا تعیین کند چه کیفیتهایی مهم هستند. این کیفیتها علاوه بر عملکرد و صحت عملکرد، نرم افزار، ویژگیهایی هستند که مفاهیم چرخه زندگی را دارند. در سالهای اخیر نشان داده شده است که چنین کیفیتهایی نظیر قابلیت تعمیر و نگهداری و حمل و نقل تأثیر چشمگیری بر هزینه چرخه حیات دارند.

ثانیاً، این چهارچوب ابزاری برای ارزیابی کمیتی از نحوهٔ توسعه و پیشرفت نسبت به اهداف کمی بدست آمده فراهم می آورد.

ثالثاً، این چهارچوب یک تقابل عمل از پرسنل QA در تلاش برای پیشرفت فراهم می‌آورد.

دست آخر اینکه، پرسنل تضمین کیفیت می توانند از معیار های کیفیت ضعیف برای کمک به شناسایی استانداردهایی که باید در آینده تقویت شوند بهره‌گیری نمایند.

بحث دقیق درمورد چهارچوب Cavano و Mccall و دیگر عوامل کیفیتی را در فصل ۱۹ آورده‌ایم. جالب است ذکر کنیم تقریباً همه جنبه‌های محاسبه از سال ۱۹۷۸ تاکنون یعی از زمانی که مک‌کال و کاوانو تحقیق اصلی شان را انجام دادند دچار تغییرات عمده‌ای شده است. ولی آن ویژگیهایی که معیار ی برای کیفیت نرم‌افزار بشمار می‌آیند هنوز تغییر نکرده‌اند.

این به چه معنی است؟ اگر یک سازمان نرم‌افزاری مجموعه‌ای از عاملهای کیفیت را بعنوان یک «فهرست بررسی (Checklist)» برای ارزیابی کیفیت نرم‌افزار بپذیرد، این احتمال وجود دارد که نرم‌افزاری که امروز ساخته می‌شود در چند دهه اولیه همین قرن دارای کیفیت و کارآیی بالایی باشند. حتی زمانی که معماری محاسبات دستخوش تغییر می‌شود، نرم‌افزاری که دارای کیفیت بالایی از لحاظ، عملکرد، انتقال و بررسی است، باز هم برای کاربران مفید و کارآمد باقی می‌ماند.

شگفت آور آنکه، فاکتورها و عوامل تعریف شده در خصوص کیفیت نرم افزار در دهه ۱۹۷۰ میلادی مشابه فاکتورهایی است که برای کیفیت نرم افزار در اوایل قرن جاری تعریف شده است.

1.McCall, J.A.

2.frameworks

۴-۵-۲ اندازه گیری کیفیت

اگر چه اندازه های فراوانی از کیفیت نرم‌افزار وجود دارد، ولی قابلیت تصحیح، قابلیت تعمیر و نگهداری، پیوستگی و قابلیت استفاده، معیار های مفیدی در اختیار تیم پروژه قرار می‌دهند. Gilb برای هریک از آنها تعاریف و اندازه‌هایی را پیشنهاد کرده است.

صحت.

یک برنامه باید بدرستی کار کند وگرنه ارزش اندکی برای کاربران خواهد داشت. قابلیت تصحیح همان میزانی است که نرم‌افزار عملکرد مورد نیازش را اجرا می‌کند. رایج‌ترین اندازه برای قابلیت اصلاح و تصحیح، تعداد نقص‌ها در هر هزار خط برنامه است، که نقص را بعنوان عدم انطباق با نیازمندیها تعریف کرده‌اند.

قابلیت نگهداری.

نگهداری مستلزم تلاش بیشتری نسبت به دیگر فعالیتهای مهندسی نرم‌افزار می باشد. قابلیت نگهداری همان سهولت تصحیح یک برنامه در هنگام مواجه شدن با یک خطا است. هیچ راهی برای اندازه‌گیری قابلیت نگهداری بطور مستقیم وجود ندارد، بنابراین باید از اندازه‌های غیرمستقیم بهره بگیریم. یک متریک ساده مبتنی بر زمان، *متوسط زمان تغییر*[1] است. یعنی زمانی که به تحلیل درخواست تغییر، طراحی یک اصلاح مناسب، پیاده سازی تغییر، آزمون آن و توزیع آن تغییر بین همه کاربران سپری خواهد شد. بطور متوسط، همه برنامه‌هایی که قابل نگهداری هستند دارای MTTC کمتری نسبت به برنامه‌هایی هستند که قابل نگهداری نیستند.

هیتاچی [TAJ81] از یک متریک هزینه برای قابلیت نگهداری استفاده کرده است که اسپویلج[2] نامیده می‌شود. یعنی هزینه تصحیح نقص‌هایی که پس از تحویل نرم‌افزار به کاربران مشاهده می‌شوند. هنگامی که نسبت خرابی به هزینه کلی پروژه بصورت یکی از عملکردهای زمان ثبت و ترسیم گردد، یک مدیر می‌تواند تعیین کند که آیا قابلیت تعمیر و نگهداری کلی نرم‌افزار تولید شده توسط یک سازمان توسعهٔ نرم‌افزار، پیشرفت داشته یا خیر، پس می‌تواند بوسیله بینش بدست آمده از این اطلاعات اقداماتی اتخاذ نماید.

جامعیت (تمامیت).

تمامیت نرم‌افزار در دوره‌ای که بطور فزاینده با مزاحمان و متجاوزان نرم‌افزارها مواجه می‌شویم، اهمیت دارد. این ویژگی توانایی یک نرم‌افزار را برای مقاومت در برابر دستبردها می‌سنجد. حمله‌ها می‌توانند بر هر سه قسمت: برنامه‌ها، داده‌ها و اسناد صورت پذیرند.

1.mean-time-to-change(MTTC)

2.spoilage

برای اندازه گیری جامعیت، دو ویژگی دیکر را باید تعریف نمائیم: تهدید[1] و امنیت[2]. تهدید احتمال بروز یک حمله از نوعی خاص و در محدودهٔ زمانی مشخص می‌باشد. امنیت احتمال آن است که نوع خاصی از حمله دفع شود. میزان تمامیت یک سیستم را می‌توان بصورت ذیل تعریف نمود:

تمامیت = Σ [(امنیت − ۱) × (تهدید − ۱)]

که تهدید و امنیت در هر نوع حمله، با یکدیگر جمع می‌شوند.

قابلیت استفاده (سهولت کاربرد).

این عبارت رایج «سهولت کاربرد» در همه بحثهای محصولات نرم‌افزاری رواج یافته است. اغلب فرض بر آن است که اگر یک برنامه دارای «سهولت کاربرد» نباشد، دچار نقص خواهد شد و بدرستی عمل نخواهد کرد، حتی درصورتیکه عملکردهای آن ارزشمند باشند.

قابلیت استفاده تلاشی برای تعیین میزان سهولت کاربرد است و بر حسب چهار ویژگی اندازه گیری می‌شود: ۱- مهارت فیزیکی یا هوشی که برای یادگیری سیستم موردنیاز است. ۲- زمان موردنیاز برای ماهرشدن در استفاده از سیستم. ۳- افزایش خالص بهره وری که موقعی اندازه‌گیری می‌شود که سیستم توسط فردی که بطور متوسط کارآیی دارد مورد استفاده قرار می‌گیرد. ۴- یک ارزیابی ذهنی و معقول از دیدگاههای کاربران نسبت به سیستم. بحث دقیق درمورد این موضوع را در فصل ۱۵ آورده‌ایم.

چهار عاملی که در بالا توصیف نمودیم فقط نمونه‌ای از عامل‌هایی هستند که به عنوان معیارهایی برای کیفیت نرم‌افزار پیشنهاد شده‌اند. در فصل ۱۹ این موضوع بطور دقیق و کامل بررسی می‌شود.

۴-۵-۳ کارآیی رفع نقص

یک متریک کیفیتی که در سطح فرآیند و پروژه مزایایی را فراهم می‌کند کارآیی رفع عیب (DRE)[3] نام دارد . در اصل، DRE یک اندازه از توانایی فیلتر کردن فعالیتهای کنترل و تضمین کیفیت در موقعی بشمار می‌آید که در همه فعالیتهای چارچوب فرآیند بکار رفته باشند.

کارآیی رفع نواقص (DRE) چیست؟

DRE را هنگامی‌که برای یک پروژه بطور کلی مورد بررسی قرار گرفته شود، بصورت ذیل تعریف می‌کنیم:

$$DRE = E / (E + D) \qquad (۴-۴)$$

که E تعداد خطاهایی است که قبل از تحویل نرم‌افزار به کاربر نهایی مشاهده شده و D تعداد خطاهایی است که بعد از تحویل یافت شده‌اند.

---

1. Threate
2. Security
3. defect removal efficiency(DRE)

ارزش و تعداد ایده‌آل برای DRE مساوی ۱ می باشد، یعنی هیچ خطایی در نرم‌افزار یافت نشود. درحالت واقعی، D بزرگتر از صفر است ولی ارزش DRE هنوز هم می‌تواند به ۱ نزدیک باشد. هنگامی که E افزایش می‌یابد، ارزش کلی DRE به ۱ می‌رسد. درواقع، هنگامی که E افزایش پیدا می‌کند، این احتمال بوجود می آید که ارزش D کاهش یابد. DRE درصورتیکه بعنوان یک متریک بکار رود که همزمان معیاری برای توانایی و قابلیت فیلتر کردنِ کنترل کیفیت و فعالیتهای تضمینی بشمار آید، تیم پروژه نرم‌افزاری را تشویق می‌کند تا فنونی را برای پیدا کردن بیشترین تعداد خطای ممکن، قبل از تحویل به کاربران، مورد استفاده قرار دهد.

کارآیی رفع نقص (DRE) را به عنوان معیاری برای کارآیی فرآیند های تضمین کیفیت به کاربرید. اگر از مراحل تحلیل و طراحی، (DRE) را یافتید، وقت بیشتری را برای بهبود فنون بازبینی رسمی سپری کنید.

DRE را می‌توان در یک پروژه برای ارزیابی توانایی تیم در پیدا کردن خطاها قبل از انتقال به وظیفه مهندسی نرم‌افزار یا فعالیت چهارچوب بعدی بکار برد. برای مثال، تحلیل نیزمندیها، یک مدل تحلیلی را بوجود می‌آورد که می‌تواند برای پیدا کردن و تصحیح خطاها مورد بررسی قرار گیرد. آن خطاهایی که طی مرور مدل تحلیلی به وظیفه طراحی منتقل می‌شوند، پیدا نخواهند شد. DRE را در این متن بصورت ذیل تعریلف می‌کنیم:

$$DRE_i = E_i / (E_i + E_{i+1}) \qquad (۵-۴)$$

که E_i تعداد خطاهایی است که طی فعالیت i ام مهندسی نرم‌افزار یافت می‌شوند، E_{i+1} تعداد خطاهایی است که طی فعالیت $i+1$ ام مهندسی نرم‌افزار یافت شده‌اند و می توانسته اند در فعالیت پیشین مهندسی نرم افزار یافت شوند (مربوط به فعالیت قبلی می باشند. م) ولی کشف نشده‌اند.

یک هدف اصلی برای یک تیم مهندسی نرم افزار (یا یک فرد مهندس نرم افزار) آن است که به DRE نزدیک به ۱ دست یابد. یعنی تمام خطاهایی که در یک فعالیتِ وجود دارد پیش از پرداختن به فعالیت بعدی مشخص و فیلتر شوند.

۶-۴ متریک های انسجام و جامعیت در فرآیند مهندسی نرم افزار

اکثر توسعه دهندگان نرم‌افزار هنوز هم اندازه گیری نمی‌کنند و متأسفانه اکثر آنها تمایل ندارند این کار را تشریح کنند. همانطور که قبلاً در همین فصل ذکر کردیم، این مشکل امری فرهنگی است. اگر فردی تلاش کند اندازه‌هایی را جمع‌آوری کند که قبلاً اصلاً جمع‌آوری نشده‌اند با مقاومت افراد روبرو خواهد شد. مثلاً یک مدیر پروژه با ناراحتی می‌پرسد : «چرا ما به این کار نیاز داریم؟»، یک فرد که در این زمینه کار می‌کند و از اضافه کاری خسته شده است با لحنی شکوه‌آمیز می‌گوید: «من نمی فهمم موضوع چیست؟»

در این بخش، چندین استدلال برای متریک ها را بررسی کرده و روشی را برای جایگزین کردن یک برنامهٔ جمع‌آوری متریک ها در یک سازمان مهندسی نرم‌افزار ارائه می‌کنیم. ولی قبل از آغاز، یکی از گفته‌های گریدی و کاسول [GRA87] را نقل می‌کنیم.

«برخی چیزهایی که در اینجا توصیف کرده‌ایم بسیار ساده بنظر می‌رسند. درواقع، تعیین یک برنامه موفقیت‌آمیز برای متریک های نرم‌افزار که در سطح کمپانی بکار گرفته شود کاری بس دشوار است. هنگامی که می‌گوئیم حداقل باید سه سال صبر کنید تا روشهای سازمانی بوجود آیند. شما تاحدی از هدف چنین تلاشی مطلع می‌شوید.»

این هشدار که توسط نویسندگان ذکر شده است جای بسی توجه دارد. ولی مزایای اندازه‌گیری آنچنان فراوان هستند که ارزش سخت کار کردن را دارند.

۴-۶-۱ استدلالی بر متریک های نرم افزاری

چرا اندازه گیری فرآیند مهندسی نرم‌افزار و محصول کاری آن چنین حائز اهمیت است؟ پاسخ معلوم است. اگر اندازه‌گیری نکنیم، هیچ راه و روشی برای ارزیابی پیشرفت وجود نخواهد داشت. و اگر درحال پیشرفت و بهبود نباشیم، گم می‌شویم.

از طریق درخواست و ارزیابی میزان بهره وری و اندازه‌های کیفی، یک مدیر ارشد می‌تواند اهداف معنی‌داری برای پیشرفت فرآیند مهندسی نرم‌افزار تعیین نماید. در فصل اول ذکر نمودیم که برای بسیاری از کمپانی‌ها نرم‌افزار یک مسأله راهبردی تجاری است. اگر فرآیند توسعه آنرا بتوانیم بهبود

بخشیم، تأثیر مستقیمی بر نتیجه حاصله خواهد داشت. ولی برای تعیین اهداف پیشرفت، باید وضعیت فعلی پیشرفت نرم‌افزار را بفهمیم. پس، اندازه‌گیری برای تعیین یک مبنای پردازشی بکار می‌رود که پیشنهاد را از روی آن بتوانیم ارزیابی کنیم.

کار سخت روزانه پروژه نرم‌افزار وقت اندکی برای فکر کردن به مسائل راهبردی باقی می‌گذارد. مدیران پروژه نرم‌افزار به مسائل اجباری‌تری می‌پردازند که عبارتند از: توسعه برآوردهای معنی‌دار برای پروژه‌ها، تولید سیستم‌های کیفیت بالاتر، تولید محصول کاملاً به موقع. از طریق استفاده از اندازه‌گیری برای تعیین مبنای پروژه، هر یک از این مسائل را می‌توان بهتر مدیریت نمود. قبلاً ذکر نمودیم که این مبنا بعنوان اساس برآوردها عمل می‌کند. افزون بر این، جمع‌آوری متریک‌های کیفیت به یک سازمان امکان می‌دهد تا بمنظور رفع دلایل نقص‌هایی که بالاترین تأثیر را بر تکمیل نرم‌افزار دارند، روند فرآیند نرم‌افزار را تنظیم نماید.[1]

نقل قول

در بسیاری از جنبه های زندگی ما مدیریت امور با استفاده از اعداد انجام می گیرد. این اعداد روشنگر و یاری دهنده اقدامات ما خواهد بود. میکائیل ماه اری پوتنام

در سطح تکنیکی و پروژه، متریک های نرم افزار، مزایای بسیار سریعی فراهم می‌کنند. هنگامی که طراحی نرم‌افزار تکمیل می‌شود، اکثر توسعه دهندگان برای یافتن پاسخ سئوالهای ذیل اضطراب دارند:

- کدامیک از تجهیزات کاربران به احتمال زیاد تغییر خواهد کرد؟
- کدامیک از پیمانه ها در این سیستم در معرض خطای بیشتر هستند؟

۱. این ایده ها به طور رسمی در یک رهیافت به نام تضمین آماری کیفیت نرم افزار قرار می گیرد و در فصل ۸ به تفصیل بحث شده اند.

- برای هر پیمانه چه تعداد آزمون باید طراحی کنیم؟
- هنگام آزمون نرم افزارهای جدید، باید انتظار چه تعداد خطا را داشته باشیم؟

درصورتیکه متریک ها جمع آوری و بعنوان یک راهنمای تکنیکی بکار رفته باشند، پاسخ این سئوالها را می‌توان تعیین نمود. در فصلهای بعد بررسی می‌کنیم که این کار چگونه انجام می‌شود.

۴-۶-۲ استقرار خط مبنا

از طریق تعیین یک مبنای متریک، می‌توان به فواید بسیاری در سطح فرآیند، پروژه و محصول دست یافت. با اینحال اطلاعاتی که جمع‌آوری می‌شود نبایستی اساساً مختلف باشد. همان متریک ها می‌توانند برای بسیاری از فایل‌های اصلی بکار گرفته شوند. مبنای متریک های حاوی اطلاعات جمع‌آوری شده از پروژه‌های قبلی توسعه نرم‌افزار است و می‌تواند مثل جدول مندرج در شکل ۴-۴ ساده باشد یا مثل یک مبنای جامع که حاوی دهها اندازه‌گیری پروژه و متریک های مشتق از آنها است، باشد.

برای مؤثر واقع شدن در بهبود فرآیند و برآورد تلاش و هزینه، داده‌های خط مبنا بایستی ویژگیهای ذیل را داشته باشند :

۱- داده‌ها باید در حد معقول صحیح باشند از «برآوردهای حدسی» درمورد پروژه‌های گذشته باید اجتناب شود.

۲- بایستی برای پروژه‌های متعدد داده‌ها را جمع‌آوری کنیم.

۳- اندازه ها باید منطبق باهم باشند، یعنی یک خط برنامه بایستی در همه پروژه‌ها کاملاً تفسیر شود.

۴- برنامه‌های کاربردی باید مشابه یا کاری باشند که برآورد شده است.

متریک ها کدام اطلاعات بحرانی را می توانند برای یک توسعه دهنده، فراهم سازند؟

۴-۶-۳ جمع آوری، محاسبه و ارزیابی متریک ها

روند تعیین یک خط مبنا را در شکل ۴-۷ نشان داده‌ایم. درحالت ایده‌آل، داده‌هایی که برای تعیین یک خط مبنا موردنیاز هستند، بطور مستمر جمع‌آوری می‌شوند. متأسفانه این کار بندرت انجام می‌شود. بنابراین جمع‌آوری داده‌ها مستلزم بررسی تاریخی پروژه‌های گذشته بمنظور بازسازی داده‌های موردنیاز می‌باشد. زمانی که اندازه‌ها جمع‌آوری می‌شوند، استفاده از متریک ها میسّر می‌شود. متریک ها می‌توانند 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 فرآیند، مورد ارزیابی قرار گیرند. ارزیابی متریک، دلایل پنهان در نتایج حاصله را روشن می‌کند و مجموعه شاخص‌هایی را برای راهنمایی پروژه یا فرآیند بوجود می‌آورد.

متریک های خط پایه می تواند از پروژه های نرم افزاری بزرگ پیشین جمع آوری شوند

۴-۷ توسعه و تکمیل متریک ها و GQM

نقل قول

اگر بنا باشد که بیایی را در کلماتی اندک برای مدیریت ارسال کنیم به آنان خواهم گفت که تغییرات و پراکندگی را کاهش دهند. دبلیو ادوارد دمینگ

یکی از مشکلاتی که [illegible] متریک ها در دو دهۀ گذشته [illegible] متریک هایی است که برای توسعه [illegible] نرم‌افزار پدید داشتند و تاریخچه‌ای از [illegible] متریک ها در محیطهای صنعتی براساس معیار سهولت استفاده، وجود دارد. در اواخر دهۀ ۱۹۸۰ [illegible] درحالیکه ارزیابی وسیعی درمورد متریک ها و مشخص ساختن اصول تئوریکی آنها صورت [illegible] بوده ولی ابزارهای لازم برای انتخاب یا تولید متریک ها توسط توسعه دهندۀ نرم‌افزار وجود نداشته است. از این بخش ترکیب GQM می‌باشد که یکی از شناخته شده ترین و پراستفاده‌ترین روش تولید متریک هایی است که توسط ویکتور باسیلی و همکارانش در دانشگاه مریلند تکمیل گردید. باسیلی و همکارانش تاریخچه‌ای طولانی از ارزیابی متریک ها در دهۀ هشتاد داشتند و GQM (اهداف، پرسش و متریک) را در آزمایشگاه مهندسی نرم‌افزار سازمان فضانوردی آمریکا NASA بوجود آوردند. باسیلی اظهار کرده که یک سازمان برای دارا بودن یک برنامه اندازه‌گیری صحیح باید این سه اجزاء را در جای خودشان داشته باشد:

۱- یک فرآیند که بتواند اهداف پروژه‌اش را ایجاد نماید.

۲- فرآیند که این اهداف را به داده‌های پروژه‌ای تبدیل نماید که این اهداف را درقالب اصطلاحات نرم‌افزار منعکس می‌سازند.

۳- فرآیندی که داده‌های پروژه را بمنظور درک اهداف تفسیر می‌نماید.

یکی از اهدافی که توسط کمپانی تعیین می‌شود این است که مقدار کار روی طراحی یک سیستم به دلیل مشکلاتی که توسط برنامه ریزان کشف می‌شود تا میزان ۸۰٪ کاهش یابد. چنین سئوال از این مثال مطرح می‌شوند که برای روشن کردن اهداف و توسعۀ متریک ها موردنیاز هستند. مجموعه‌ای از این سئوالهای نمونه را در ذیل می‌آوریم:

- قصد دارید چگونه مقدار کار مجدد را تعیین نمائید؟
- آیا اندازۀ محصول باید در محاسبه کار مجدد مورد استفاده قرار گیرد؟
- آیا افزایش کار لازم برای رازیابی و طراحی دوباره را در مقایسه با فرآیند کنونی و ارزیابی یک طرح درنظر گرفته‌اید؟

نقل قول

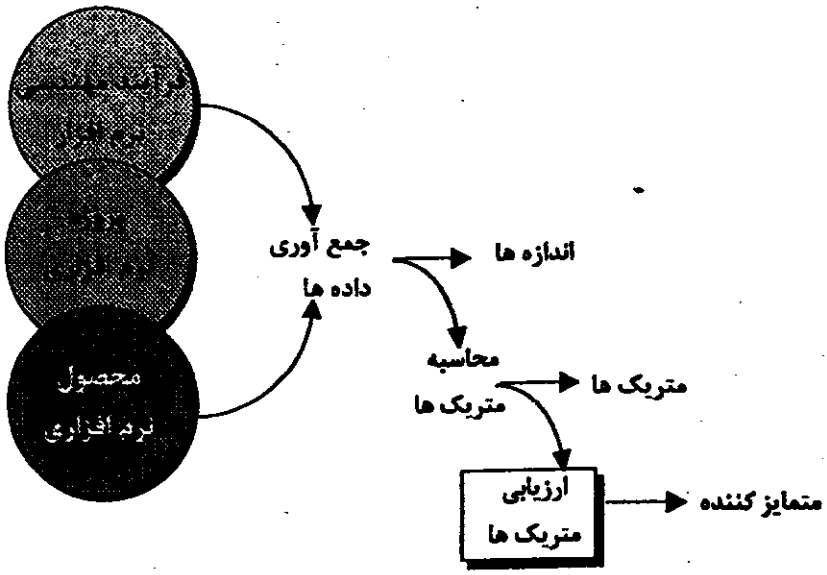
چگونه می توانیم از اعتبار آماری متریک‌های گردآورده شده اطمینان حاصل نماییم؟

نقل قول

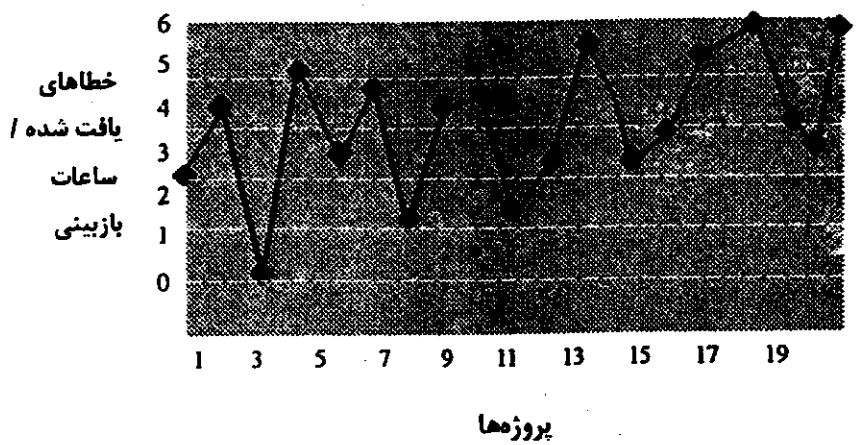
اگر سیگنال های خطا نباشد، از کجا خواهیم دانست که تغییرات فرآیند در جهت بهبود بوده است یا عقبگرد؟ ریچارد زولتز

زمانی که این سئوالات مطرح می‌شوند، متریک هایی را می‌توان توسعه داد که اهداف را برآورده سازند و طبیعتاً از این سئوالات ناشی می‌شوند. اهمیت GQM در این حقیقت نهفته است که یکی از تلاشهای اولیه برای توسعه مجموعه‌ای از اندازه‌گیریها می‌باشد که برای نرم‌افزار قابل استفاده است و به الگوی پیشرفت نرم‌افزاری مربوط می‌شود که قبلاً بحث نمودیم. باسیلی یک الگوی پیشرفت کیفیت را تکمیل نموده که روش GQM بخوبی در آن جای می‌گیرد. این الگو شامل سه مرحله می‌باشد:

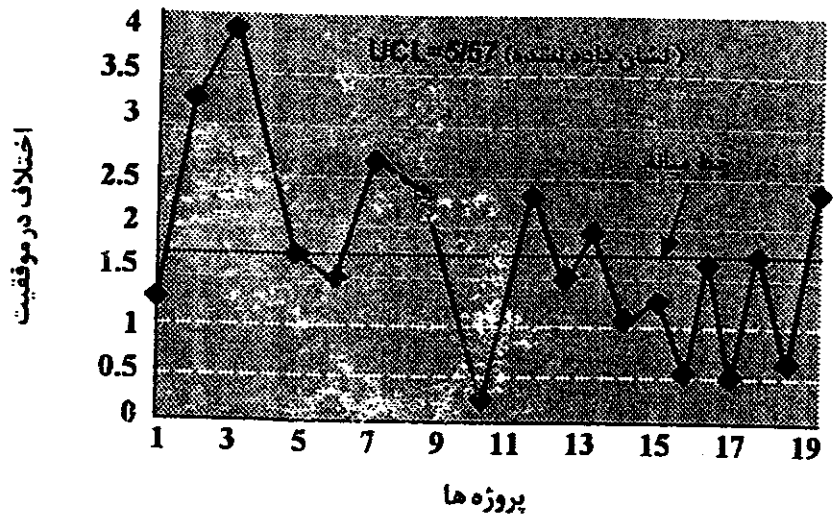
- فرآیند انجام برآورد در مورد یک پروژه و محیط آن، تعیین اهداف کیفیتی برای پروژه و انتخاب ابزارهای مناسب، روشهای مدیریتی و فنآوریهایی برای آن اهداف تا شانس برآورده شدن را داشته باشند.
- فرآیند اجرای یک پروژه و نظارت بر داده‌های مربوط به این اهداف کیفیتی این کار در رابطه با فرآیند عملکرد براساس داده‌ها درموقع برآورده نشدن اهداف کیفیتی توسط مدیر پروژه انجام می‌شود.
- فرآیندی برای تحلیل داده‌های جمع‌آوری شده در مرحله دوم بمنظور ارائه پیشنهاد برای پیشرفت بیشتر این فرآیند مستلزم جستجو بدنبال مشکلات در جمع‌آوری داده‌ها و مشکلات در بکارگیری راهنمایی‌های کیفیتی و مشکل در تفسیر داده‌ها می‌باشد.



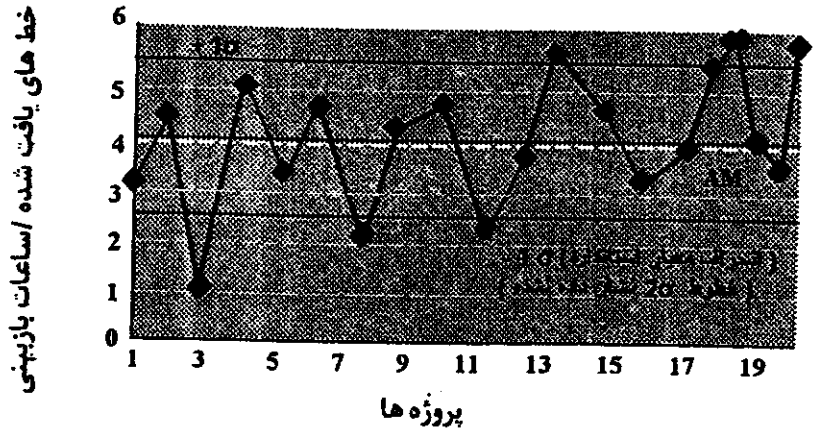
شکل ۴ـ۷ فرآیند جمع آوری متریک های نرم افزاری



شکل ۴ـ۸ داده های متریک برای خطاهای پوشش داده نشده در هر ساعت بازبینی



شکل ۴ـ۹ چارت کنترل رنج حرکت



شکل ۴ـ۱۰ چارت کنترل

باسیلی، مجموعه روشهایی را پیشنهاد کرده که برای توسعه‌دهندگانی که می‌خواهند از GQM برای توسعهٔ متریک های واقعی در پروژه‌هایشان بهره‌گیری نمایند مفید واقع می‌شود. اهداف GQM را می‌توان بوسیله سه مجموعه بازسازی نمود که هدف، جنبه و محیط را دربر می‌گیرد. مجموعه هدف برای ساختن چیزی که تحلیل شده و برای مقصود بکار می‌رود. برای مثال، یک توسعه دهنده ممکن است بخواهد مؤثر بودن بررسی‌های طرح را با هدف بهبود نرخ رفع خطای فرآیند تحلیل نماید و توسعه دهنده ممکن است بخواهد استانداردهای بکار رفته توسط این کمپانی را با هدف کاهش میزان کار در طی مراحل نگهداری مورد تحلیل قرار دهد.

۳۰۸ مدیریت تغییرات : کنترل فرآیند آماری

از آنرو که مدیریت فرآیند نرم افزار و محصول ساخته شده، هر دو از پارامترهای بسیاری تاثیر پذیر می باشند (دانش سطح دارایی متخصصین" ساختار تیم نرم افزاری، دانش مشتریان، فنآوری پیاده سازی شده، ابزارهی که در [illegible] مورد استفاده واقع شده است)، متریک های جمع آوری شده برای یک پروژه یا محصول نمی توانند با متریک های جمع شده برای دیگر پروژه ها سنخوانی کامل داشته باشد. در حقیقت، تغییرات [illegible] متریک هایی که به صورت روتین در فرآیند نرم افزاری جمع می شوند، وجود دارد.

از آنرو که متریک های فرآیند از پروژه ای به پروژه دیگر تغییر خواهد داشت، چگونه میتوانیم بگوییم که متریک ها بهبود یافته اند یا تنزل داشته اند؟ چگونه می توانیم به روش آماری اطمینان داشته باشیم و آن را حاصل یک اختلال ندانیم؟ تغییرات معیارهای خاص نرم افزاری (مثبت یا منفی) چگونه معنادار خواهند بود؟

یک تکنیک گرافیکی برای تعیین محفاظر بودن داده های متریکی مربوط به تغییرات وجود دارد. نمودار کنترل نامیده می شود و توسط والتر شوارت در دهه ۱۹۲۰ ابداع شده است. این تکنیک افراد علاقه مند به بهبود فرآیند نرم افزاری را قادر می سازد که تعیین کند آیا پراکندگی (تغییرات) و مکان (میانگین حرکت) متریک های فرآیند پایدار می باشند (فرآیند تنها تغییرات طبیعی یا کنترل شده را از خود بروز می دهد) یا غیر پایدار (فرآیند تغییرات غیرقابل کنترلی دارد و متریک ها نمی توانند عملکرد دقیق را نشان دهند). دو نوع متفاوت از نمودارهای کنترل برای ارزیابی داده های متریک ها مورد استفاده دارند [ZUL99] : ۱) نمودار کنترل طیف حرکت ۲) نمودار کنترل فردی(مستقل).

برای توضیح رهیافت نمودار کنترل، یک سازمان نرم افزاری را در نظر گیرید که متریک های فرآیند را شامل تعداد خطاهای کشف شده در هر ساعت کار بازبینی، E_r ، جمع آوری می کند. پس از ۱۵ ماه، سازمان برای ۲۰ پروژه کوچک در حوزه مشابه توسعه نرم افزار، مقدار E_r ، را جمع آورده است. مقادیر به دست آمده برای E_r ، در شکل ۸.۴ نشان داده شده اند. E_r از مقدار اندک ۱/۲ برای پروژه شماره ۳ تا مقدار عظیم ۵/۹ برای پروژه ۱۷ متغیر بوده است. طی تلاشی که برای بهبود کارآمدی بازبینی ها ، سازمان نرم افزار تهیه دید، آموزش و نظارت بر تمام اعضاء تیم ها صورت پذیرفت، این امر از پروژه ۱۱ آغاز شد.

ریچارد زولتنر رویه ای کلی برای توسعه یک نمودار کنترل طیف حرکتی (mR) تهیه دیده است که پایداری فرآیند را تعیین می نماید [ZUL99] :

۱- طیف های حرکت را محاسبه کنید : قدر مطلق [illegible] اختلاف متوالی میان هر جفت نقاط داده... این نقاط طیف حرکتی را روی نمودار خود رسم نمایید.

۲- میانگین طیف حرکت را محاسبه کنید... این (میله "mR") را به عنوان خط مرکزی در نمودار خود رسم کنید.

۳- میانگین را در عدد ۳/۲۶۸ ضرب نمایید. این خط را به عنوان حد بالای کنترل رسم کنید. (UCL) این خط به اندازه سه انحراف استاندارد بالای خط مرکزی قرار دارد.

با استفاده از داده های مشهود در شکل ۴-۸ و مراحل توصیه شده توسط زولتنر یا یکه نمودار کنترل mR را توسعه داده ایم که در شکل ۴-۹ دیده می شود. مقدار خط mR (میانگین) برای داده های طیف حرکتی ۱/۷۱ می باشد. حد کنترل بالایی نیز ۵/۵۸ می باشد.

برای تعیین این امر که پراکندگی متریک های فرآیند پایدار می باشد، یک سوال ساده پرسیده خواهد شد: آیا تمام مقادیر طیف حرکتی در پایین UCL قرار دارد؟ برای مثال مورد نظر ما پاسخ "بله" می باشد. بنابرین پراکندگی متریک ها پایدار می باشد.

نمودار (چارت) کنترل آنرا نیز به طریقه زیر ساخته می شود:

۱- مقادیر متریک های فردی را مانند آنچه در شکل ۴-۸ دیده شد، رسم کنید.

۲- برای مقادیر متریک ها، A_m مقدار متوسط را محاسبه کنید.

۳- مقدار میانگین برای مقادیر mR را (مورد mR) در عدد ۳/۲۶۸ ضرب کنید و مقدار A_m به دست آمده در مرحله قبل را با آن جمع کنید. این نتایج حد بالای فرآیند طبیعی (UNPL) خواهند بود. آن را رسم کنید.

۴- مقدار میانگین برای مقادیر mR را (مورد mR) در عدد ۳/۲۶۸ ضرب کنید و مقدار A_m به دست آمده در مرحله قبل را از آن کم کنید. این نتایج حد پایین فرآیند طبیعی (LNPL) خواهند بود. آن را رسم کنید. اگر LNPL کمتر از ۰/۰ بود، نیازی به رسم آن نیست، تا آنکه متریک های مورد ارزیابی مقادیری کمتر از ۰/۰ پیدا نمایند.

۵- انحراف استاندارد را از فرمول $(UNPL - A_m)/3$ به دست آورید. خطوطی را در یک یا دو برابری انحراف استاندارد رسم نمایید. اگر انحراف استاندارد کمتر از ۰/۰ وجود داشت، نیازی به رسم آن نیست، تا آنکه متریک های مورد ارزیابی مقادیری کمتر از ۰/۰ پیدا نمایند.

این گامها برای داده های نشان داده شده در شکل ۴-۸ بکار رفته است، و ما همانطور که در شکل ۴-۱۰ دیده می شود، یک نمودار کنترل فردی را به دست آورده ایم.

زولتنر [ZUL99] چهار معیار اصلی را مورد بازبینی قرار داد، که آنها را *قواعد زون* می نامند. این معیارها نشان می دهند که آیا تغیرات نشان داده شده توسط متریک ها بیانگر تحت کنترل بودن یا خارج از کنترل بودن فرآیند می باشند. اگر هریک از شرایط زیر " صحیح " باشد، داده‌های متریک نشانگر خارج از کنترل بودن فرآیند می باشد:

۱- یک مقدار منفرد متریک ها خارج از UNPL.

۲- قرار داشتن دو سوم مقادیر متوالی متریک ها، در فاصله بیش از دو برابر انحراف استاندارد از A_m.

۳- قرار داشتن چهار پنجم مقادیر متوالی متریک ها، در فاصله بیش از یک برابر انحراف استاندارد از A_m.

۴- قرار داشتن هشت مقدار متوالی متریک ها، در یک سوی A_m.

از آنرو که هیچیک از این شرایط برای مقادیر نشان داده شده در شکل ۴ـ۱۰ صحیح نمی باشد، داده های متریک ما از یک فرآیند پایدار به دست آمده اند و اطلاعات روند به روشنی از متریک های جمع آوری شده قابل تشخیص است. با ارجاع به شکل ۴ـ۱۰ دیده می شود که تغییرات E_r پس از پروژه ۱۰ کاهش یافته است(پس از کوشش و تلاشی که برای بهبود بازبینی ها صورت پذیرفته است). با محاسبه مقدار متوسط ۱۰ پروژه اول و ۱۰ پروژه آخر نشان داده می شود که مقدار متوسط E_r برای پروژه های ۱۱ تا ۲۰ نسبت به پروژه های ۱ تا ۱۰ ، ۲۹ درصد بهبود و پیشرفت داشته است. در حالیکه نمودار کنترل نشان می دهد فرآیند پایدار می باشد، تلاش صورت پذیرفته برای بهبود تاثیر بازبینی ها ، موفقیت آمیز بوده است.

۴-۹ متریک های سازمان های کوچک

تعداد بسیاری از سازمان های توسعه دهنده نرم افزار، کمتر از ۲۰ نفر کارمند دارند. غیر مستدل و برخی موارد غیر واقع بینانه است که از چنین سازمانهایی انتظار داشته باشیم برنامه های متریک نرم افزاری قابل اعتنایی داشته باشند. با این وجود منطقی است که به چنین سازمانهایی توصیه کنیم تمام اندازه های مستقیما" اندازه گرفتنی خصوصا مبتنی بر سایز را جمع آوری کرده با متریک های بدست آمده در پیشرفت فرآیند محلی توسعه نرم افزار خود بکوشند و کیفیت و زمان آماده سازی محصول را بهبود بخشند. کاتز [KAU99] یک سناریو را که نوعاً هنگام پیشنهاد برنامه های متریک به سازمان های کوچک روی می دهد،اینگونه بیان می دارد :

در ابتدای امر تولید کنندگان نرم افزار به فعالیتهای ما به چشم بدبینی و تردید نگاه می کردند، اما بالاخره آن را مورد پذیرش قرار دادند و این از آن جهت بود که اندازه ها را ساده، مناسب با هر سازمانی تنظیم می کردیم و آنها اطمینان حاصل نمودند که اطلاعات با ارزشی را در اختیار خواهند داشت.

سرانجام، برنامه های مراقبت از مشتریان، طرح ریزی و برنامه ریزی برای آینده ارائه گردید.

آنچه کاتز پیشنهاد می کند رهیافتی مبتنی بر عقل سلیم جهت پیاده سازی هر فعالیت مرتبط با فرآیند نرم‌افزاری می باشد: آن را ساده بگیرید، برای نیازهای محلی آن را تنظیم نمایید، و مطمئن باشید که ارزش افزوده خواهد داشت. در پاراگراف بعدی ما این رهنمودها را در خصوص متریک های فروشگاه های کوچک به کار خواهیم بست.

" ساده انگاشتن " رهنمودی است که در بسیاری از فعالیت ها نتیجه مطلوب خواهد داشت. اما چگونه متریک های نرم‌افزای ساده را به دست آوریم که ارزشمند نیز باشند، و چگونه مطمئن باشیم که این متریک های ساده نیازهای یک سازمان نرم افزاری خاص را برآورده می کنند؟ ما با تکیه بر نتایج آغاز خواهیم کرد و البته نه اندازه گیری ها. گروه نرم افزاری باید هدف یکتایی را که نیاز به بهبود دارد، تعیین نماید. برای مثال "کاهش زمان ارزیابی و پیاده سازی درخواست های تغییرات". یک سازمان نرم افزاری کوچک باید مجموعه اندازه های زیر را که به سادگی جمع شدنی هستند، انتخاب نماید:

- زمانی که سپری می شود(به ساعت یا روز) تا یک درخواست داده شده، کاملا" ارزیابی شود، t_{queue}
- نیروی کار مورد نیاز (برحسب نفر – ساعت) برای ارزیابی، W_{eval}.
- زمانی که سپری می شود (بر حسب ساعت یا روز) تا سفارش تغییر یک ارزیابی کامل شده به پرسنل داده شود، t_{eval}.
- نیروی مورد نیاز برای اعمال تغییرات (نفر – ساعت)، W_{change}.
- زمان مورد نیاز (ساعت یا روز) 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}.

هنگامی که این اندازه ها برای تعدادی از درخواست های تغییر جمع گردید، زمان کل سپری شده از درخواست تغییر تا پیاده سازی تغییر قابل محاسبه خواهد بود و درصد زمان هایی که به انتظار اولیه گذشته و یا ارزیابی و تخصیص تغییر و پیاده سازی تغییر طول کشیده، محاسبه می شود. به طور مشابه درصد نیرویی که برای ارزیابی و پیاده سازی مورد نیاز است نیز، می تواند تعیین شود. این متریک ها در قالب داده های کیفی قابل ارزیابی می باشند، اندازه هایی چون E_{change} و D_{change} . درصدها، در خصوص آهستگی فرآیند درخواست تغییر آگاهی لازم را ایجاد نموده و در خصوص گامهای پیشرفت و بهبود فرآیند، راهنما خواهند بود، این امر با کاهش t_{queue} و W_{eval} و t_{eval} و W_{change} و / یا E_{change} انجام خواهد شد. بعلاوه کارآیی رفع نقص به قرار ذیل قابل محاسبه می باشد :

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

DRE می تواند با زمان سپری شده و کل نیروی کار مقایسه شود و تاثیر فعالیت های تضمین کیفیت بر زمان و نیروی کار لازم برای اعمال تغییرات تعیین گردد.

برای گروه های کوچک هزینه جمع آوری اندازه ها و محاسبه متریک ها بین ۳ تا ۸ درصد بودجه پروژه را طی مرحله آموزش به خود اختصاص خواهد داد، و بعد از آشنا شدن مدیران پروژه و مهندسین نرم افزار با برنامه متریک ها، این میزان به کمتر از ۱ درصد کل بودجه پروژه کاهش خواهد یافت [GRA99].

در صورتی که داده های متریک ما در بهبود معنادار فرآیند سازمان نرم افزاری بصیرتی را بوجود آورند، این هزینه ها بازگشت سرمایه قابل توجهی را نشان خواهند داد.

۴-۱۰ برقراری یک برنامه برای متریک های نرم افزاری

انستیتو مهندسی نرم افزار برای برقراری یک برنامه متریک های نرم افزاری " مبتنی بر هدف " کتابی راهنما [PAR96] منتشر نموده است. کتاب راهنمای مذکور گامهای زیر را توصیه می نماید:

۱- اهداف تجاری خود را تعریف کنید.

۲- مشخص نمایید که چه چیزهایی را می خواهید بدانید یا فرا بگیرید.

۳- [illegible] نوعی خود را مشخص نمایید.

۴- موجودیتها و صفات خاصه مرتبط با [illegible] نوعی را تعریف کنید.

۵- [illegible] اندازه گیری خود را رسمی نمایید.

۶- پرسشهای کمی و شاخصهای مرتبط را مشخص نمایید، همانهایی که شما را در دستیابی به اهداف اندازه گیری یاری می نمایند.

۷- عناصر داده ای را مشخص کنید، آنهایی که شما را برای یافتن پاسخهای مناسب جهت پرسشهایتان کمک خواهند نمود.

۸- اندازه های مورد استفاده را مشخص کنید، این تعاریف را عملیاتی کنید.

۹- اقداماتی که برای پیاده سازی این اندازه گیری ها باید انجام دهید، [illegible] ص کنید.

۱۰- یک طرح و برنامه برای پیاده سازی اندازه گیری ها آماده کنید.

توضیح کامل این گامها در کتاب راهنما آمده است. با این وجود یک دید خلاصه از نکات کلیدی ارزشمند خواهد بود.

از آنرو که نرم افزار کارکردهای تجاری را پشتیبانی می کند، تفاوت میان سیستمهای مبتنی بر کامپیوتر با محصولات را نمایان می کند و یا خود به عنوان یک محصول عمل می کند، اهداف تعریف شده تجاری تقریبا" همواره در سطح مهندسی نرم افزار نیز اهداف مشخصی خواهند بود. برای مثال در نظر آورید که شرکتی سازنده سیستمهای امنیتی پیشرفته باشد که نرم افزارهای اساسی و مهمی را شامل است. مهندسین نرم افزار و مدیران تجاری به عنوان یک تیم کاری، لیستی از اهداف تجاری اولویت بندی شده را ارائه خواهند نمود:

۱- رضایت مشتریان را نسبت به محصولاتمان افزایش دهیم.

۲- استفاده از محصولاتمان را ساده تر سازیم.

۳- زمان لازم برای ارائه محصول جدید به بازار را کاهش دهیم.

۴- پشتیبانی محصولاتمان را ساده تر و روان تر انجام دهیم.

۵- سودکلی را افزایش دهیم.

سازمان نرم افزاری، هریک از اهداف تجاری را بررسی کرده و خواهد پرسید : " چه فعالیتهایی را انجام می دهیم یا مدیریت می کنیم و چه کنیم که این فعالیتها بهبود یابند؟ " برای پاسخ به این سوالات موسسه مهندسی نرم افزار یک "لیست موجودیت – پرسش " را توصیه می کند که در آن همه چیز (تمام موجودیتها) که توسط فرآیند مهندسی نرم افزار مدیریت می شوند و از آن تاثیر می پذیرند، مورد توجه سازمان نرم افزاری خواهند بود. مثالهایی از موجودیت ها عبارتند از منابع توسعه، محصولات کاری، کد منبع (برنامه) ، موارد آزمون، درخواست تغییرات، وظائف مهندسی نرم افزار، و زمان بندی ها. برای هر فعالیت لیست شده، افراد نرم افزاری مجموعه ای از پرسشها را توسعه می دهند که به ارزیابی کمی خصوصیات یک موجودیت منجر خواهد شد.(مانند اندازه، هزینه و زمان توسعه). ایجاد لیست موجودیت – پرسش به تهیه پرسشهایی منجر می شود که مجموعه اهداف فرعی مرتبط به موجودیت ها را به دست می دهد و در ضمن فعالیتهای انجام شده به عنوان بخشی از فرآیند نرم افزار محسوب خواهند شد.

هدف چهارم را در نظر آورید :" پشتیبانی محصولاتمان را ساده تر انجام دهیم." برای این هدف می توان مجموعه پرسشهای زیر را مطرح نمود [PAR96] :

- آیا درخواستهای تغییری که از سوی مشتری رسیده است، حاوی اطلاعات دقیق برای ارزیابی نحوه اعمال و پیاده سازی آن تغییرات و تعیین زمانی آن می باشد؟
- درخواست های تغییرات عقب افتاده چه اندازه است؟
- آیا زمان پاسخ دهی ما به رفع و رجوع خطاها همان قدر است که مشتری انتظار دارد؟
- آیا ما فرآیند کنترل تغییرات را (فصل ۹) دنبال می کنیم؟
- آیا تغییراتی که از اولویت بالایی برخوردارند در زمان مناسب پیاده می شوند؟

بر پایه این پرسشها، سازمان نرم افزاری هدف فرعی زیر را به دست خواهد آورد : *پیشرفت عملکرد فرآیند مدیریت تغییر*. موجودیتها و صفات خاصه فرآیند نرم افزاری مرتبط با هدف فرعی تعریف شده و اهداف اندازه گیری مربوط به آنها تعیین می شوند.

موسسه مهندسی نرم افزار[PAR90] برای رهیافت اندازه گیری مبتنی بر هدف گامهای ۶ تا ۱۰ ، راهنمایی مفصل‌تر را تهیه نموده است. در حقیقت، یک فرآیند پالایش مرحله ای انجام می گیرد که طی آن اهداف به پرسشها پالایش شده و پرسشها نیز به موجودیتها و صفات خاصه تبدیل می شوند و آنها نیز به نوبه خود به متریک ها پالایش خواهند شد.

۴-۱۱ خلاصه

اندازه گیری، مدیران و متخصصین را قادر می سازد که فرآیند ساخت نرم افزار را بهبود بخشند. آنانرا در برنامه ریزی و ردگیری و کنترل پروژه های نرم افزاری یاری می کند و کمک می کند که نسبت به کیفیت محصول نرم افزاری تولید شده ارزیابی داشته باشند. معیارهای ویژگیهای خاص فرآیند، پروژه و محصول برای محاسبه متریک های نرم افزاری کاربرد خواهند داشت. این متریک ها تحلیل شده و شاخص هایی را به دست می دهند که مدیریت را در انجام اقدامات فنی یاریگر خواهند بود.

متریک های فرآیند، سازمان را یاری می کنند که به یک دید راهبردی دست یابد، دیدگاهی که به درک موثر بودن و کارآمدی یک فرآیند نرم افزاری کمک می کند. متریک های پروژه تاکتیکی می باشند. و مدیران پروژه را یاری می کنند که جریان کار پروژه را و رهیافت فنی را با زمان واقعی تطبیق دهند.

متریک های مبتنی بر سایز و مبتنی بر کارکرد هردو، در صنعت کاربرد دارند. متریک های مبتنی بر اندازه تعداد خطوط برنامه را برای نرمال سازی دیگر عوامل مانند نفر – ماه یا تعداد نواقص و عیبها ، ... کار می گیرند. امتیاز کارکردی از اندازه زدن اطلاعات حوزه و ارزیابی موضوعی پیچیدگی مسئله به دست می آید.

متریک های کیفیت نرم افزار، مانند متریک های بهره وری، بر فرآیند ، پروژه و محصول متمرکز می شوند. با توسعه و تحلیل یک خط مبنای برقرار شده برای متریک ها جهت کیفیت، یک سازمان می تواند فرآیند نرم افزاری را آنگونه تغییر دهد که مشکلات و نواقص رفع شوند.

متریک ها تنها وقتی معنا دار و معتبر می باشند که اعتبار آماری آنها بررسی و تایید شده باشد. نمودار کنترل، شیوه ساده‌ای برای نیل به این هدف است و همزمان بررسی تغییرات و تعیین نتایج متریک ها توصیه می شود.

اندازه گیری منتج به تغییرات فرهنگی خواهد شد. جمع آوری داده ها، محاسبه متریک ها، و تحلیل آنها سه گامی است که باید برای شروع شدن برنامه متریک ها برداشته شوند. به طور کلی، یک رهیافت مبتنی بر هدف، به سازمان کمک می کند که بر متریک های معتبر متمرکز شود. با برقراری و ایجاد یک خط مبنای متریک، - یک پایگاه داده ها که اندازه های مربوط به محصول و فرآیند را در بر دارد – مهندسین نرم افزار و مدیرانشان می توانند نسبت به آن کاری که انجام می دهند و آن محصولی که توسط ایشان ساخته می شود بصیرت و آگاهی بیشتری داشته باشند.

مسایل و نکاتی برای تفکر و تعمق بیشتر

۴-۱ سه اندازه، سه متریک و شاخص‌های مربوط به آن را پیشنهاد کنید که در ارزیابی یک خودرو به کار روند.

۴-۲ سه اندازه، سه متریک و شاخص‌های مربوط به آن را پیشنهاد کنید که در ارزیابی دپارتمان خدمات یک بنگاه خودرو مورد استفاده داشته باشند.

۴-۳ اختلاف میان متریک های فرآیند و پروژه را با زبان خودتان شرح دهید.

۴-۴ چرا برخی متریک های نرم‌افزاری را باید «خصوصی» دانست؟ سه مثال از متریک هایی بیاورید که باید خصوصی بمانند و سه مثال از آنهایی که باید به‌اطلاع عموم برسند.

۴-۵ یک نسخه از هامفری (مقدمه ای بر فرآیند نرم افزار شخصی، ادیسون ویزلی، ۱۹۹۷) بدست آورید و به طور خلاصه‌در یک یا دو صفحه در خصوص رهیافت PSP بنویسید.

۴-۶ گریدی برای متریک های نرم‌افزاری آدابی پیشنهاد می‌کند. آیا می‌توانید سه قاعده به قواعد ذکر شده در بخش ۴-۲-۱ بیافزایید؟

۴-۷ کوشش کنید نمودار استخوان ماهی شکل ۴-۳ را تکمیل کنید. به این صورت که با دنبال کردن رهیافت به‌کار رفته برای مشخصه های «غیر صحیح» ، اطلاعات قابل مقایسه ای برای مشخصه های «از قلم افتاده، مبهم و تغییر یافته» فراهم کنید.

۴-۸ اندازه غیرمستقیم چیست و چرا چنین اندازه هایی در کنار متریک های نرم‌افزاری متداولند؟

۴-۹ تیم الف پیش از تحویل محصول و در طی فرآیند مهندسی نرم‌افزار ۳۴۲ خطا یافته است. تیم ب ۱۸۴ خطا یافته است. برای این‌که تعیین کنیم کدام تیم خطاها را به صورت کارآتری رفع نموده است، چه اندازه های دیگری از دو پروژه الف و ب باید منظور نظر قرار گیرد؟ برای تعیین این امر چه متریک هایی را در نظر دارید؟ چه اطلاعات تاریخی ممکن است مفید باشند؟

۴-۱۰ در مقابل متریک تعداد خطوط برنامه جهت ارزیابی بهره وری نرم افزار و در نقد آن نظری و استدلالی ارائه کنید. آیا استدلال شما برای یک دوجین یا صدها پروژه نیز معتبر می باشد؟

۴-۱۱ مقدار امتیاز عملکرد را برای پروژه‌ای با خصوصیات حوزه اطلاعاتی زیر محاسبه کنید:

تعداد ورودی‌های کاربر: ۳۲

تعداد خروجی‌های کاربر: ۶۰

تعداد پرس و جوهای کاربر: ۲۴

تعداد فایل‌ها: ۸

تعداد رابط های خارجی: ۲

فرض کنید همهٔ مقادیر تنظیم پیچیدگی، در سطح متوسط باشند.

۴-۱۲ مقدار امتیاز عملکرد سه‌بعدی را برای یک سیستم درون نهاده (تعبیه شده) با خصوصیات زیر محاسبه کنید:

ساختمان داده داخلی: ۶

ساختمان داده خارجی: ۳

تعداد ورودی‌های کاربر: ۱۲

تعداد خروجی‌های کاربر: ۶۰

تعداد پرس و جوهای کاربر: ۹

تعداد رابط‌های خارجی: ۳

تبدیلات: ۳۶

انتقال ها (گذرها): ۲۴

فرض کنید پیچیدگی این شمارش‌ها به طور مساوی بین کم، میانگین و زیاد تقسیم شده اند.

۴-۱۳ نرم‌افزاری که برای کنترل یک‌دستگاه فتوکپی استفاده می شود، به ۳۲۰۰۰ خط C و ۴۲۰۰ خط اسمالتک نیاز دارد. تعداد امتیازات عملکردی را برای نرم‌افزار داخل دستگاه برآورد کنید.

۴-۱۴ مک‌کال و کاوانو (بخش۴-۵-۱) یک «چارچوب» برای کیفیت نرم‌افزار تعریف کرده‌اند. با استفاده از اطلاعات این کتاب و دیگر کتابها، هر یک از سه «نقطه نظر» را به مجموعه‌ای از متریک ها و عوامل کیفیتی توسعه دهید.

۴-۱۵ (غیر از آنهایی که در این کتاب عرضه شده اند) متریک هایی را خود برای درستی و صحت ، قابلیت نگهداری، جامعیت و یکپارچگی و قابلیت استفاده توسعه دهید. اطمینان حاصلب کنید که آنها قابل ترجمه به مقادیر کمّی می باشد.

۴-۱۶ آیا امکان دارد همزمان با کاهش تعداد نواقص و عیوب در هر هزار خط برنامه، ضایعات افزایش یابد؟توضیح دهید.

۴-۱۷ آیا هنگام استفاده از فنون نسل چهارم اندازه تعداد خطوط(LOC) معنایی خواهد داشت؟ توضیح دهید.

۴-۱۸ یک سازمان نرم‌افزاری، داده‌های کارآیی رفع خطا DRE را برای ۱۵ پروژه طی دوسال گذشته در اختیار دارد. مقادیر جمع شده عبارتند از: ۰/۸۱، ۰/۷۱، ۰/۸۷، ۰/۵۴، ۰/۶۳، ۰/۷۱، ۰/۹۰، ۰/۸۲، ۰/۶۱ ، ۰/۸۴، ۰/۷۳، ۰/۸۸، ۰/۷۴، ۰/۸۶، ۰/۸۳ . نمودارهای کنترل مستقل و mR را ایجاد کنید و تعیین کنید که آیا می توان از این داده‌ها برای ارزیابی روندها استفاده کرد؟

فهرست منابع و مراجع

[ALA97] Alain, A., M. Maya, J.M. Desharnais, and S. S1. Pierre, "Adapting Function Points to Real-Time Software," *American Programmer*, vol. 1O, no. 11, November 1997, pp. 32-43.

[ALB79] Albrecht, A.J., "Measuring Application Development Productivity," *Proc. IBM Application Development* Symposium, Monterey, CA, October 1979, pp. 83-92.

[ALB83] Albrecht, A.J. and J.E. Gaffney, "Software Function, Source Lines of Code and Development Effort Prediction: A Software Science Validation," *IEEE Trans. Software Engineering*, November 1983, pp. 639-648.

[ART85] Arthur, *L.J., Measuring Programmer Productivity and Software Quality*, Wiley-Interscience, 1985.

[BOE81] Boehm, B., *Software Engineering Economics*, Prentice-Hall, 1981.

[GRA87] Grady, RB. and D.L. Caswell, *Software* Metrics: *Establishing a Company-wide Program*, Prentice-Hall, 1987.

[GRA92] Grady, RG.. *Practical Software* Metrics *for Project Management and Process Improvement*, Prentice-Hall, 1992.

[GRA94] Grady, R, "Successfully Applying Software Metrics," *Computer*, vol. 27, no. 9, September 1994, pp. 18-25.

[GRA99] Grable, R, et aI., "Metrics for Small Projects: Experiences at SED," *IEEE Software*, March 1999, pp. 21-29.

[GIL88] Gilb, T., *Principles of Software Project Management*, Addison-Wesley, 1988.

[HET93] Hetzel, W., *Making Software Measurement Work*, QED Publishing Group, 1993.

[HUM95] Humphrey, W., *A Discipline for Software* Engineering, Addison-Wesley, 1995.

[IEE93] *IEEE Software Engineering Standards*, Standard 610.12-1990, pp. 47-48.

[IFP94] Function *Point Counting Practices Manual*, Release 4.0, International Function Point Users Group, 1994.

(JON86] Jones, c., *Programming Productivity*, McGraw-Hili, 1986.

(JON91] Jones, c., *Applied Software Measurement*, McGraw-Hili, 1991.

(JON98] Jones, c., *Estimating Software Costs*, McGraw-Hili, 1998.

[KAU99] Kautz, K., "Making Sense of Measurement for Small Organizations," *IEEE Software*, March 1999, pp. 14-20.

[MCC78] McCall, J.A. and J.P. Cavano, "A Framework for the Measurement of Software Quality," ACM Software Quality Assurance Workshop, November 1978.

[PAR96] Park, RE., W.B. Goethert, and W.A. Florac, *Goal Driven Software Measurement-A Guidebook*, CMU/SEI-96-BH-002, Software Engineering Institute, Carnegie Mellon University, August 1996.

[PAU94] Paulish, D. and A. Carleton, "Case Studies of Software Process Improvement Measurement," *Computer*, vol. 27, no. 9, September 1994, pp. *50-57.*

[RAG95] Ragland, B., "Measure, Metric or Indicator: What's the Difference?" *Crosstalk*, vol. 8, no. 3, March 1995, p. 29-30.

[TAJ81] Tajima, D. and T. Matsubara, "The Computer Software Industry in Japan," *Computer*, May 1981, p. 96.

[WHI95] Whitmire, S.A., "An Introduction to 3D Function Points", *Software* Development, April 1995, pp. 43-53.

[ZUL99] Zultner, RE., "What Do Our Metrics Mean?" *Cutter IT journal*, vol. 12, no. 4,April1999,pp.II-19.

خواندنیهای دیگر و منابع اطلاعاتی

Software process improvement (SPI) has received a significant amount of attention over the past decade. Since measurement and software metrics are key to successfully improving the software process, many books on SPI also discuss metrics. Worthwhile additions to the literature include:

Burr, A. and M. Owen, *Statistical Methods for Software Quality*. International Thomson Publishing, 1996.

El Emam, K. and N. Madhavji (eds.), *Elements of Software Process Assessment and Improvement*, IEEE Computer Society, 1999.

Florac, W.A. and A.D. Carleton, *Measuring the Software Process: Statistical Process Control for Software Process Improvement*, Addison-Wesley, 1999.

Garmus, D. and D. Herron, *Measuring the Software Process: A Practical Guide to Functional Measurements*, Prentice-Hall, 1996.

Humphrey, w., *Introduction to the Team Software Process*, Addison-Wesley Longman, 2000.

Kan, S.H., *Metrics and Models in Software Quality Engineering*, Addison-Wesley, 1995.

Humphrey [HUM95]. Yeh *(Software Process Control*, McGraw-Hili, 1993). Hetzel [HET93]. and Grady [GRA92] discuss how software metrics can be used to provide the indicators necessary to improve the software process. Putnam and Myers *(Executive Briefing: Controlling Software Development*, IEEE Computer Society, 1996) and Pulford and his colleagues (A *Quantitative Approach to Software Management*, Addison-Wesley, 1996) discuss process metrics and their use from a management point of view. Weinberg *(Quality Software Management*, Volume 2: *First Order Measurement*, Dorset House, 1993) presents a useful model for observing software projects, ascertaining the meaning of the observation, and determining its significance for tactical and strategic decisions. Garmus and Herron *(Measuring the Software Process*, Prentice-Hall, 1996) discuss process metrics with an emphasis on function point analysis. The Software Productivity Consortium *(The Software Measurement Guidebook*, Thomson Computer Press, 1995) provides useful suggestions for instituting an effective metrics approach. Oman and Pfleeger *(Applying Software Metrics*, IEEE Computer Society Press, 1997) have edited an excellent anthology of important papers on software metrics. Park, et at. [PAR96] have developed a detailed guidebook that provides step-by-step suggestions for instituting a software metrics program for software process improvement.

The newsletter *IT Metrics* (edited by Howard Rubin and published by Cutter Information Services) presents useful commentary on the state of software metrics in the industry. The magazines *Cutter IT journal* and *Software Development* have regular articles and entire features dedicated to software metrics.

A wide variety of information sources on software process and project metrics are available on the Internet. An up-to-date list of World Wide Web references that are relevant to the software process and project metrics can be found at the SEPA Web site:

bttp://www.mhhe.com/ engcs/ compsci/pressman/resources/ process-metrics.mhtmI