فصل ۱۹ متریک های فنی نرم افزار

مفاهیم کلیدی (مرتب بر حروف الفبا)

اصول اندازه گیری ، صفات خاصه متریک ها ، عوامل کیفیت ، متریک های آزمون ، متریک های برنامه منبع ، متریک های تحلیل ، متریک های مشخص سازی ، متریک های معماری

KEY CONCEPTS

analysis metrics , architectural metrics , component-level metrics , design metrics , maintenance metrics , measurement principles , metrics attributes , quality factor , source code metrics , testing metrics

نگاه اجمالی

متریک های فنی نرم افزار چیست؟ ماهیتاً مهندسی یک دسیپلین کمیتی است. مهندسین از اعداد برای کمک به طراحی استفاده کرده و محصول ساخته شده را ارزیابی می‌کنند. تا همین اواخر، مهندسین نرم‌افزار از رهنمودهای اندکی در کار خود برخوردار بودند. اما این مسئله در حال تغییر است. سیستم متریک فنی دوباره به مهندسین کمک می‌کند تا از طراحی و ساخت محصول خود شناخت حاصل کنند.

چه کسی عهده دار انجام آن است؟ مهندسین نرم‌افزار از سیستم متریک فنی برای ایجاد نرم‌افزار کیفی بالاتر استفاده می‌کنند.

چرا این مسئله از اهمیت برخوردار است؟ همواره یک عنصر مقداری در ایجاد نرم‌افزار کامپیوتر وجود دارد. مشکل آن است که ارزیابی کمیتی، شاید کافی نباشند. یک مهندس نرم‌افزار به معیارهایی نیاز دارد که به او در طراحی داده‌ها، معماری، تعامل و اجزاء یاری کند. آزمون‌کننده به راهنمایی مقداری که به انتخاب موارد آزمون و اهداف آن یاری رساند، نیاز دارد. سیستم متریک فنی، پایه‌ای را برای تحلیل، طراحی برنامه نویسی و آزمون ارائه می‌کند که این آزمون به طور ملموس تری انجام شده و از نظر کمیتی بهتر ارزیابی شود.

چه مراحلی پیش رو هستند؟ اولین مرحله در فرآیند اندازه‌گیری، کسب مقیاس‌های نرم‌افزار و متریک مناسب برای باز نمود نرم‌افزار در نظر گرفته شده است. مرحلهٔ بعدی، داده‌های مورد نیاز برای دستیابی به سیستم متریک فرموله شده است. زمانی‌که محاسبه گردید، متریک مناسب بر پایه رهنمودهای از پیش مقرر شده، تحلیل شده و داده‌های گذشته در نظر گرفته می شوند. نتایج تحلیل تفسیر شده تا

شناخت بهتری از کیفیت نرم‌افزار حاصل شود و نتایج تفسیر مربوطه منتهی به اصلاح محصولات برآمده از تحلیل، طراحی، کد و یا آزمون می‌شود.

محصول کار چیست؟ سیستم متریک نرم‌افزاری که از طریق محاسبهٔ داده‌های جمع‌آوری شده از تحلیل و مدل‌های طراحی، برنامه منبع و موارد آزمون، بدست می‌آیند.

چگونه مطمئن شوم که کار را به‌درستی انجام داده‌ام؟ شما باید اهداف اندازه‌گیری را قبل از جمع‌آوری داده‌ها، تعیین کرده و هر متریک فنی را به‌گونه‌ای واضح تعریف کنید. فقط تعداد محدودی متریک را تعریف کرده و بعد آنها را برای حصول شناخت از کیفیت محصول کار مهندسی نرم‌افزار به‌کار ببرید.

یک عنصر کلیدی هر فرآیند مهندسی، اندازه‌گیری است. ما از مقیاس‌ها برای فهم بهتر صفات‌خاصه مدل‌هایی که بوجود آورده‌ایم و برای ارزیابی کیفیت محصولات و یا سیستم‌های ساخته شده بدست خود، استفاده می کنیم. ولی برخلاف سایر اصول مهندسی، مهندسی نرم‌افزار ریشه در قوانین کمیتی فیزیک نداشته و مقیاس‌های مطلق، مانند ولتاژ، جرم، شتاب یا درجه حرارت در این دنیای نرم‌افزاری متعارف نیستند. در عوض ما تلاش می‌کنیم تا نشانه‌ای از کیفیت برخی ویژگی‌های نرم‌افزار ارائه کنیم. از آن‌جا که مقیاس‌ها و سیستم متریک نرم‌افزار مطلق نیستند. می‌توان دربارهٔ آنها بحث کرد. فنتون [FEN91][1] این‌گونه به این موضوع می‌پردازد:

اندازه‌گیری، فرآیند اختصاص نمادها و شماره‌ها به صفات‌خاصه موجودیتها در دنیای واقعی است به‌گونه‌ای که مطابق با قوانین به‌خوبی تعریف شده، تعریف شوند ... در علوم فیزیکی، پزشکی، اقتصاد و اخیراً علوم اجتماعی، ما می‌توانیم صفات‌خاصه اندازه‌گیری را که پیش از این قابل اندازه‌گیری نمی‌دانستیم را اکنون، اندازه‌گیری کنیم.

البته این‌گونه اندازه‌گیری به اندازهٔ دیگر اندازه‌گیری‌ها در علوم فیزیکی، پالایش نشده‌اند اما آنها وجود دارند (و تصمیمات مهمی بر پایه آنها اتخاذ می‌شود) ما احساس می‌کنیم که مجبور به تلاش ناگزیر برای اندازه‌گیری هر آن‌چه قابل اندازه‌گیری نیست، هستیم تا فهم بهتری از موجودیت های خاص بدست آوریم. این امر جایگاه بسیار قدرتمندی در مهندسی نرم‌افزار دارد.

اما تعدادی از اعضای جامعه نرم‌افزاری هم‌چنان استدلال می‌کنند که نرم‌افزار قابل اندازه‌گیری نبوده و تلاش‌های مربوط به این‌کار باید تا هنگامی‌که شناخت بهتری از نرم‌افزار و صفات‌خاصه‌ای که برای توصیف آن باید به‌کار بروند، بدست نیامده است، به تعویق افتد. این یک اشتباه است.

هر چند متریک‌های فنی برای نرم‌افزار کامپیوتر مطلق نیستند. یک راه نظام مند ارزیابی کیفیت بر پایه مجموعه‌ای از قوانین به‌خوبی تعریف شده، در اختیار ما قرار می‌دهند. آنها هم‌چنین یک شناخت عینی

1.Fenton,N.

در همان لحظه را در مقایسه با پس از رخداد آن در اختیار مهندس نرم‌افزار قرار می‌دهند. این امر مهندس را قادر می‌سازد تا مشکلات را کشف کرده و آنها را قبل از آن‌که فاجعه ببار آورند، بر طرف سازند.۷

در فصل ۱۴، ما دربارهٔ متریک‌های نرم‌افزاری به همان صورت که در فرآیند سطح پروژه به‌کار می‌روند، بحث کردیم. در این فصل، توجه ما معطوف به اندازه‌هایی می‌شود که می‌توانند برای ارزیابی کیفیت محصول در حین مهندسی و ساخت آن، به‌کار روند. این مقیاس‌های به صفت‌خاصه یک نشانهٔ زمان واقعی از کارآیی تحلیل، طرح و مدل‌های برنامه نویسی، تأثیر موارد آزمون، کیفیت کلی نرم‌افزار در دست ساخت، در اختیار مهندس نرم‌افزار قرار می‌دهد.

۱۹-۱ کیفیت نرم افزار

حتی دل‌زده‌ترین طراحان نرم‌افزار تأیید می‌کنند که نرم‌افزار با کیفیت بالا یک هدف مهم است. ولی ما کیفیت[1] را چگونه تعریف می‌کنیم. در فصل ۸، ما راههای مختلفی را برای در نظر گرفتن کیفیت نرم‌افزار پیشنهاد کرده و کیفیت را معرفی کردیم که بر روی هماهنگی داشتن با نیازمندیهای صریحاً اظهار شدهٔ عملکردی و اجرایی تأکید داشت که شامل استانداردهای توسعهٔ مستند شده و خصوصیات تلویحی که از تمامی نرم‌افزارهای به‌طور حرفه‌ای طراحی شده نیز می‌گردید.

نقل قول

هر برنامه ای از قسمتهای درستی تشکیل شده است، اما همان قسمتها نیز معلوم نیست آنگونه که ما می خواهیم عمل نماید. نویسنده ناشناس

تردید کمی وجود دارد که تعریف فوق بتواند اصلاح شده، تعمیم یافته و یا تا ابد روی آن بحث شود. در رابطه با مقصود این کتاب، تعریف فوق بر روی سه نکتهٔ مهم تأکید می‌کند:

۱- نیازمندیهای نرم‌افزاری پایه سنجش کیفیت است. فقدان هماهنگی با نیازمندیها، فقدان کیفیت است.

۲- استانداردهای مشخص شده، یک مجموعهٔ معیارهای توسعه را که اندازه و مقیاس موردنظر برای مهندسی نرم‌افزار می‌باشد را تعریف می‌کند. چنانچه این معیارها رعایت نشوند، فقدان کیفیت به‌طور قطع پیش خواهد آمد.

۳- مجموعه‌ای از نیازمندیهای غیر صریح وجود دارد که اغلب ذکر نشده باقی می‌ماند. (مثلاً طلب ایجاد راحتی برای کاربر). چنانچه نرم‌افزار ما با نیازمندیهای صریح خود، هماهنگ باشد ولی نتواند نیازهای غیر صریح خود را برآورده سازد، کیفیت نرم‌افزار مورد تردید است.

کیفیت نرم‌افزار یک مخلوط پیچیده از فاکتورها است که در پهنهٔ کاربردهای مختلف، تغییر می‌کنند و مشتریان طالب آنها نیز شامل آن می‌شوند. در بخش های آینده، فاکتورهای کیفیت نرم‌افزار تعریف شده و فعالیت‌های لسانی مورد نیاز برای دستیابی به آنها شرح داده خواهد شد.

1.quality

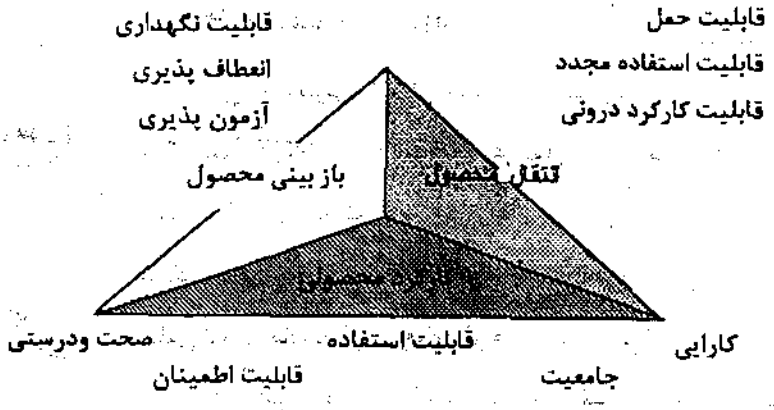
۲.توجه به این نکته مهم است که کیفیت به ویژگیهای فنی تحلیل، طراحی و مدلهای برنامه نویسی گسترش می یابد. مدلهایی که کیفیت بالا به دنبال دارند (از بعد فنی) نرم افزار را آنگونه هدایت می کنند که از نقطه نظر مشتری دارای کیفیت بالا باشد.

۱۹-۱-۱ فاکتور های کیفیت " مک کال "

فاکتورهای مؤثر در کیفیت نرم‌افزار را می‌توان به دو گروه عمده تقسیم کرد:

۱- فاکتورهایی که می‌توان آنها را مستقیماً اندازه گرفت (مثلاً نواقص در هر کارکرد - امتیاز).

۲- فاکتورهایی که فقط به‌طور غیرمستقیم اندازه‌گیری می‌شوند. (مثلاً قابلیت استفاده یا قابلیت نگهداری) در هر حالت، باید اندازه‌گیری انجام شود. ما باید نرم‌افزار را (اسناد، برنامه، داده‌ها) با تعدادی از داده‌ها مقایسه کرده و به یک دلالت کیفیتی برسیم.



شکل ۱۹ـ۱ فاکتورهای کیفیت نرم‌افزار مک‌کال

جالب است بدانید که فاکتورهای کیفیت " مک کال " که اکنون نیز از اعتبار برخوردارند، نخستین بار در دهه ۱۹۷۰ تعیین گردیدند. این خود دلیلی است که عوامل موثر بر کیفیت نرم افزار تغییری پیدا نکرده اند.

مک کال و همکارانش [MCC77][1] یک گروه‌بندی مفید برای فاکتورهایی که کیفیت نرم‌افزار را تحت تأثیر قرار می‌دهند، پیشنهاد می‌کنند. این فاکتورهای کیفیت نرم‌افزار، نمایش داده شده در شکل ۱۹-۱، بر روی سه جنبهٔ مهم یک محصول نرم‌افزاری، توجه خود را متمرکز می‌کند: خصوصیات عملیاتی، توانایی تغییر یافتن و تطبیق با محیط‌های جدید، با عطف به فاکتورهای ذکر شده در شکل ۱۹-۱ مک کال شرح ذیل را ارائه می‌کند:

درستی[2]. آن اندازه که برنامه‌ای مشخصه های خود و نیازمندیهای مربوط به مشتری را برآورده می‌کند.

قابلیت اطمینان[3]. آن اندازه که از برنامه‌ای بتوان اجرای کارکردهای مربوط به خود را با دقت لازم، انتظار داشت. (باید خاطرنشان کرد تعاریف کامل‌تر دیگری دربارهٔ قابلیت اطمینان، پیشنهاد شده‌اند (به فصل ۸ مراجعه کنید).

1.McCall,J.
2.Correctness
3.Reliability

کارآیی[1]. مقدار منابع محاسبه و کد مورد نیاز برای آن‌که برنامه‌ای کارکرد خود را اجرا کند.

جامعیت[2]. اندازه‌ای که افراد غیر مجاز را می‌توان از دستیابی به نرم‌افزار یا داده‌ها منع کرد.

قابلیت استفاده[3]. تلاش مورد نیاز برای یادگیری، عملیات، آماده کردن ورودی و تفسیر خروجی یک برنامه.

قابلیت نگهداری[4]. تلاش مورد نیاز برای شناسایی مکان یک اشتباه در برنامه و رفع کردن آن (این یک تعریف خیلی محدود شده است).

قابلیت انعطاف[5]. تلاش مورد نیاز برای اصلاح یک برنامه عملیاتی.

آزمون پذیری[6]. تلاش لازم برای آزمودن برنامه جهت حصول از اجرای کارکرد تعیین شده.

قابلیت حمل[7]. تلاش مورد نیاز برای انتقال برنامه از یک سخت‌افزار و/ یا محیط سیستم نرم‌افزار به محیطی دیگر.

قابلیت استفاده مجدد[8]. آن اندازه که برنامه‌ای (یا قسمت‌های یک برنامه) را بتوان در دیگر کاربردهای دوباره به‌کار برد. مرتبط با بسته‌بندی و میزان عمل کارکردهایی که برنامه اجرا می‌کند.

قابلیت درون عملیاتی[9]. تلاش مورد نیاز برای تلفیق سیستمی به سیستم دیگر.

نقل قول

کیفیت یک محصول بستگی به چگونگی رفتار آن در قبال تغییرات جهان واقعی دارد. دمارکو

بسیار مشکل و گاهی اوقات غیرممکن است که بتوان فاکتورهای کیفیت فوق را با مقیاس‌های مستقیم اندازه‌گیری کرد. بنابراین، یک مجموعه‌ای از متریک‌ها تعریف شده و به‌کار می‌رود تا بتوان هر یک از فاکتورها را مطابق رابطهٔ ذیل، ارائه کرد:

$$F_q = c_1 \times m_1 + c_2 \times m_2 + \ldots + c_n \times m_n$$

در این رابطه F_q فاکتور کیفیت نرم‌افزار، c_n ضریب رگرسیون، m_n متریک‌هایی است که روی فاکتور کیفیت تأثیر می‌گذارد. متأسفانه، بسیاری از متریک‌های تعریف شده توسط مک کال را فقط می‌توان به‌صورت ذهنی و موضوعی اندازه‌گیری کرد. این متریک‌ها می‌توانند به‌صورت یک چک لیست برای

1.Efficiency
2.Integrity
3.Usability
4.Maintainability
5.Flexibility
6.Testability
7.Portability
8.Reusability
9.Interoperability

"درجه‌بندی" صفات خاصه ویژه نرم‌افزار، باشند [CAV78][1]. شمای درجه‌بندی توصیه شده توسط مک کال چنین است. مقیاس ۰ (پایین) تا ۱۰ (بالا)، متریک‌های ذیل در شمای درجه‌بندی به‌کار می‌روند.

متریک هایی که امکان ارزیابی را در بازبینی های مبتنی بر فنون رسمی فراهم می سازند، در فصل ۸ توضیح داده شده اند.

قابلیت وارسی[2]. راحتی کنترل رعایت شدن استانداردها.

دقت[3]. دقیق بودن محاسبات و کنترل

رایج بودن ارتباطات[4]. میزان تعامل‌های استاندارد، پروتکل‌ها و میزان استفاده گسترده.

تکمیل بودن[5]. میزان پیاده‌سازی تکمیل کارکرد موردنظر.

فشرده بودن[6]. فشردگی برنامه از لحاظ خطوط کد.

سازگاری[7]. به‌کارگیری طراحی یک شکل و فنون مستندسازی در سرتاسر پروژهٔ توسعهٔ نرم‌افزار.

رایج بودن داده‌ها[8]. به‌کارگیری انواع و ساختارهای داده‌های استاندارد در سرتاسر برنامه.

فاکتورهای کیفیت

تولارانس اشتباه[9]. خسارت وارده هنگام برخورد برنامه با اشتباه.

کارآیی اجرایی[10]. عملکرد زمان اجرای یک برنامه.

قابلیت توسعه[11]. میزان قابلیت تعمیم داده‌ها، معماری و یا طراحی رویه‌ها.

عمومیت داشتن[12]. گستردگی کاربرد بالقوهٔ اجزاء برنامه.

استقلال سخت‌افزاری[13]. میزان جدایی نرم‌افزار از سخت‌افزاری که روی آن عمل می‌کند.

به‌کارگیری ابزار[14]. میزان نظارت برنامه بر عملیات خود و شناسایی اشتباهات رخ داده.

پیمانه ای بودن[15]. استقلال کارکردی (فصل ۱۳) اجزاء برنامه.

قابلیت اجرا[1]. راحتی عملیات برنامه.

1.Cavano,J.P.
2.Auditability
3.Accuracy
4.Communication commonality
5.Completeness
6.Conciseness
7.Consistency
8.Data commonality
9.Error tolerance
10.Execution efficiency
11.Expandability
12.Generality
13.Hardware independence
14.Instrumentation
15.Modularity

امنیت[2]. فراهم بودن مکانیسم‌های کنترل و حفاظت برنامه‌ها و داده‌ها.

خود ـ مستندسازی[3]. میزان برخوردار بودن برنامه منبع از مستندارت معنادار.

سادگی[4]. میزان قابل فهم بودن برنامه بدون شکل.

استقلال سیستم نرم‌افزار[5]. میزان استقلال یک برنامه از ویژگی‌های زبان برنامه‌سازی غیراستاندارد، خصوصیات سیستم عملیاتی، و دیگر محدودیت‌های محیطی.

قابلیت ردیابی[6]. توانایی ردیابی یک بازنمایی طراحی یا جزء عملی برنامه، از نیازمندیها.

آموزش[7]. میزان کمک نرم‌افزار به توانا کردن کاربران جدید به اجرای سیستم.

رابطه بین فاکتورهای کیفیت نرم‌افزار و متریک‌های فهرست شده در فوق در شکل ۱۹-۲ نمایش داده شده است. باید خاطرنشان کرد که وزن داده شده به متریک وابسته به محصولات و ملاحظات محلی است.

۱۹-۱-۲ FURPS

فاکتورهای کیفیت تشریح شده توسط مک کال و همکارانش [MCC77][8] نمایانگر یکی از مجموعه "چک لیست"های کیفیت نرم‌افزار، پیشنهاد شده است. هیولت ـ پاکارد [GRA87] مجموعه‌ای از فاکتورهای کیفیت نرم‌افزار ارائه داده‌اند که یک اسم اکرونیم به آن اطلاق شده است: FURPS. قابلیت کارکردی، قابلیت به‌کارگیری، قابلیت اطمینان، اجرا و قابلیت پشتیبانی، فاکتورهای کیفیت FURPS. به‌طور آزادانه از کارهای قبلی استفاده کرده و صفات خاصه ذیل را بر هر یک از ۵ فاکتور عمده تعریف می‌کند:

- **قابلیت کارکردی**[9] از طریق ارزیابی مجموعه ویژگیها و توانایی‌های برنامه، عمومیت کارکردهایی که تحویل داده می‌شوند و امنیت کل سیستم، سنجیده می‌شود.
- **قابلیت به‌کارگیری**[10] با لحاظ کردن فاکتورهای انسانی (فصل ۵)، زیبایی‌شناس، سازگاری و مستندسازی جامع، سنجیده می‌شود.

1.Operability
2.Security
3.Self-documentation
4.Simplicity
5.Software system independence
6.Traceability
7.Training
8.McCall,J.
9.Functionality
10.Usability

- **قابلیت اطمینان**[1] از طریق اندازه‌گیری فرکانس و شدت شکست، توانایی بهبود شکست، دقت نتایج خروجی، زمان میانگین شکست (MTTF) و قابل پیش‌بینی بودن برنامه ارزیابی می‌شود.
- **عملکرد**[2] از طریق سرعت پردازش، زمان پاسخ، مصرف منابع، توان‌عملیاتی و کارآیی سنجیده می‌شود.
- **قابلیت پشتیبانی**[3] توانایی تعمیم برنامه (قابلیت تعمیم)، قابلیت تطبیق، قابلیت خدماتی (این خصلت نمایان‌گر یک واژهٔ رایج‌تر یعنی نگهداری است) را با یکدیگر ترکیب می‌کند، به‌علاوه قابلیت آزمون پذیری، سازگار بودن، پیکربندی شدن (توانایی سازماندهی و کنترل عناصر پیکربندی نرم‌افزار، فصل ۹)، راحتی نصب سیستم و راحتی یافتن مشکلات سیستم را نیز با حاصل ترکیب قبلی، تلفیق می‌سازد.

فاکتورهای کیفیت FURPS و صفات‌خاصه تشریح شدهٔ فوق می‌توانند برای برقراری متریک‌های کیفیت هر مرحله در فرآیند مهندسی نرم‌افزار، به‌کار روند.

1.Reliability
2.Performance
3.Supportability

عوامل کیفیت / متریک های کیفیت نرم افزار	درستی	قابلیت اطمینان	کارایی	جامعیت	قابلیت نگهداری	انعطاف پذیری	آزمون پذیری	قابلیت حمل	قابلیت استفاده	درون عملیاتی	قابلیت کاربرد
قابلیت وارسی				*			*				
دقت		*									
رایج بودن ارتباطات										*	
تکمیل بودن	*										
پیچدگی		*				*	*				
فشردگی			*		*	*					
سازگاری	*	*			*	*					
رایج بودن دادها										*	
تولارانس خطا		*									
کارآیی اجرا			*								
قابلیت گسترش						*					
عمومیت داشتن						*		*	*	*	
استقلال سخت افزاری								*	*		
به کار گیری ابزار				*	*		*				
پیمانه ای بودن		*			*	*	*	*	*	*	
قابلیت اجرا			*								*
امنیت				*							
خود - مستندسازی					*	*	*	*	*		
سادگی		*			*	*	*				
استقلال سیستمی								*	*		
قابلیت ردیابی	*										
آموزش											*

(Adapted from Arthur, L. A., Measuring Productivity and software Quality, Wiley Intescience, 1985.)

شکل ۱۹ـ۲ فاکتورهای کیفیت و متریک ها

نقل قول

هر فعالیتی می تواند یک فعالیت خلاق باشد، به شرط آنکه انجام دهنده آن به صحت عمل و انجام بهتر آن بیاندیشد.

جان آپدایک

۱۹-۱-۳ فاکتورهای کیفیت ایزو ۹۱۲۶

استاندارد ایزو ۹۱۲۶ در تلاش برای تشخیص خصلت‌های کلیدی کیفیت در نرم‌افزار کامپیوتری، ارائه شد. این استاندارد شش خصلت کیفیت کلیدی را تعیین می‌کند:

- **قابلیت کارکردی**[1]. میزان برآورده کردن نیازهای اظهار شده آن‌گونه که از طریق خصیصه‌های فرعی ذیل، بیان شده است: مناسب بودن، دقیق بودن، درون ـ عملیاتی بودن، متابعت و امنیت.
- **قابلیت اطمینان**[2]. مقدار زمانی‌که در دسترس نرم‌افزار است تا مطابق با آن‌چه در خصلت‌های فرعی ذیل بیان شده است، مورد استفاده قرار دهد: پختگی، تولارانس خطا، قابلیت بهبود و بازیابی.
- **قابلیت به‌کار گرفته شدن**[3]. میزان راحتی استفاده از نرم‌افزار آن‌گونه که توسط این خصلت‌های فرعی بیان شده است: قابلیت ادراک شدن، قابلیت یادگیری آن، قابلیت اجرای عملیات روی آن.
- **کارآیی**[4]. میزان به‌کارگیری بهینهٔ نرم‌افزار از منابع سیستم آن‌گونه که توسط خصلت‌های فرعی ذیل ارائه شده است: رفتار زمانی، رفتار منبعی.
- **قابلیت نگهداری شدن**[5]. راحتی تعمیر کردن نرم‌افزار آن‌گونه که خصلت‌های فرعی زیر شرح می دهند: قابلیت تحلیل شدن، قابلیت تغییر کردن، ثبات، قابلیت آزمون پذیری.
- **قابلیت حمل شدن**[6]. راحتی حمل و نقل نرم‌افزار از محیطی به محیط دیگر آن‌گونه که خصلت‌های فرعی ذیل بیان می‌کنند: قابلیت تطبیق، قابلیت نصب، متابعت، قابلیت جابجایی.

مانند دیگر فاکتورهای کیفیت مورد بحث قرار گرفته در بخش‌های ۱۹-۱-۱ و ۱۹-۱-۲ فاکتورهای ایزو ۹۱۲۶ضرورتاً موجب فراهم شدن اندازه‌گیری مستقیم نمی‌شوند. در هر صورت، آنها یک پایهٔ ارزشمند برای مقیاس‌های غیرمستقیم و چک لیست فوق‌العاده برای ارزیابی کیفیت یک سیستم، فراهم می‌سازند.

۱۹-۱-۴ انتقال به دیدگاه کمی

در بخش‌های گذشته، مجموعه‌ای از فاکتورهای مقداری برای "اندازه‌گیری" کیفیت نرم‌افزار مورد بحث قرار گرفتند. ما تلاش داریم تا مقیاس‌های دقیقی را برای نرم‌افزار کیفی به‌وجود آوردیم، با این‌وجود

1.Functionality
2.Reliability
3.Usability
4.Efficiency
5.maintainability
6.Portability

گاهی اوقات ماهیت موضوعی (و نه روشن و ملموس) کار، باعث دلسردی ما می‌شود. کاوانو و مک کال [CAV76][1] این وضعیت را مورد بحث قرار می‌دهند:

تعیین کیفیت یک فاکتور کلیدی در وقایع روزانه است ـ مسابقات مزه کردن شراب، وقایع ورزش (مثلاً ژیمناستیک) مسابقات استعداد و غیره. در این موقعیت‌ها کیفیت در بنیادی‌ترین و مستقیم‌ترین شکل خود، مورد قضاوت قرار می‌گیرد.

مقایسه اشیاء در کنار یکدیگر و در شرایط یکسان با مفاهیم از پیش تعیین شده. شراب (نوشیدنی م.) می‌تواند بر اساس روشنی، رنگ، رایحه، طعم و غیره، مورد قضاوت قرار گیرد. در هر حال، این نوع قضاوت بسیار موضوعی و ذهنی (در مقابل عینی ـ م.) است. این‌که آیا هیچ ارزشی دارد یا خیر باید توسط یک کارشناس روشن شود.

ذهنی و موضوعی بودن و تخصصی شدن نیز در تشخیص کیفیت نرم افزار، دخیل است. برای آن‌که به حل مشکل کمک کنیم، یک تعریف دقیق‌تر از کیفیت نرم‌افزار همراه با طریقه‌ای برای بدست آوردن اندازه‌های مقداری کیفیت نرم‌افزار جهت تحلیل ملموس و مشهود مورد نیاز است... از آن‌جا که چیزی مانند دانش مطلق وجود خارجی ندارد، نباید انتظار تعیین دقیق کیفیت نرم‌افزار را داشت. چرا که هر اندازه‌گیری تا حدودی ناقص است. ژاکوب برونکورسکی[2] این پارادکس دانش را به این طریقه شرح داده است: "هر ساله ما ابزارهای دقیق‌تری را به‌وجود می‌آوریم تا از طریق آنها طبیعت را بهتر مشاهده کنیم. زمانی‌که به مشاهدات خود نظر می‌افکنیم از این‌که آنها را هم‌چنان غیر واضح و مبهم می‌بینیم، ناراحت می‌شویم و احساس می‌کنیم که هم‌چنان در ابهام باقی مانده‌اند."

در بخش های آینده، ما مجموعه‌ای از متریک‌های نرم‌افزاری را که می‌تواند در ارزیابی کمیتی کیفیت نرم‌افزار به‌کار بروند، مورد آزمون قرار می‌دهیم. در تمام موارد، متریک‌ها نمایان‌گر مقیاس‌های غیرمستقیم هستند یعنی آن‌که ما هرگز کیفیت را اندازه نمی‌گیریم بلکه در واقع به دنبال نمودهای کیفیت هستیم. فاکتور غامض‌کنندهٔ امور رابطه مستقیم بین متغیری که باید اندازه‌گیری شود و کیفیت نرم‌افزار می‌باشد.

۱۹-۲ یک چارچوب برای متریک های فنی نرم افزار

همان‌طور که در مقدمهٔ این فصل خاطرنشان کردیم، اندازه‌گیری، اعدادی را به نمادهای صفات‌خاصه موجودیتهای جهان واقعی تخصیص می دهد. برای انجام این کار یک مدل اندازه‌گیری در بردارندهٔ یک مجموعه سازگار از قوانین مورد نیاز است. هر چند، تئوری اندازه‌گیری مثلاً [KYB84][3] و کاربرد آن در

1.Cavano,J.P.

2. Jacob Bronkowski

3.Kyburg,H.E.

نرم‌افزار کامپیوتر مثلاً، [DEM81][1] و [BRI96][2]و[ZUS97][3] موضوعاتی فراتر از میدان عمل این کتاب هستند. ایجاد یک چارچوب بنیادی و یک مجموعه از اصول اساسی برای اندازه‌گیری متریک‌های فنی نرم‌افزار، ارزشمند است.

۱۹-۲-۱ چالش متریک های فنی

طی سه دههٔ گذشته، تحقیق‌گران بسیاری تلاش کرده‌اند تا یک متریک منفرد فراهم‌کنندهٔ یک مقیاس فراگیر از پیچیدگی نرم‌افزار را ارائه نمایند. فنتون [FEN94][4] این تحقیق را این‌گونه شرح می دهد، "مقدس ناممکن". هر چند که تعدادی از مقیاس‌های پیچیدگی پیشنهاد شده‌اند [ZUS90][5]، هر کدام نقطه‌نظر متفاوتی از مفهوم پیچیدگی داشته و این‌که چه خصلت‌هایی به پیچیدگی منتهی می‌شود.

با این حال، نیاز به مقیاس و کنترل پیچیدگی نرم‌افزار وجود دارد. و اگر دستیابی به یک مقدار منفرد از این متریک کیفیت مشکل است، ایجاد مقیاس‌هایی با خصلت‌های برنامه‌ای درونی متفاوت باید ممکن باشد. (مثلاً، پیمانه سازی مؤثر، استقلال کارکردی، و دیگر خصلت‌های بحث شده در فصل های ۱۳ تا ۱۶). این مقیاس‌ها و متریک‌های حاصل از آنها را می‌توان به‌عنوان نشانه‌هایی از کیفیت تحلیل و مدل‌های طراحی، مورد استفاده قرار داد. اما در این‌جا مجدداً مشکلات بروز می‌کنند. فنتون [FEN94][6] این مطلب را چنین بیان می‌کند:

خطر کوشش در راه یافتن مقیاس‌هایی که خصلت‌های بسیار مختلفی را عرضه می‌کند، آن است که بطور غیرقابل اجتناب مقیاس مجبور از تحقق هدف‌های متناقض خواهند بود. این امر مخالف تئوری باز نمود اندازه‌گیری است.

هر چند گفتهٔ فنتون صحیح است، افراد زیادی استدلال می‌کنند که اندازه‌گیری فنی انجام شده در طول مراحل آغازین فرآیند نرم‌افزار یک مکانیسم سازگار و عینی جهت ارزیابی کیفیت در اختیار مهندس نرم‌افزار قرار می‌دهد.

ارجاع به وب

اطلاعات مفصلی در خصوص متریک های فنی توسط هورست زوئس گردآوری شده است:
www.irb.cs.tu-berlin.de/~zuse/

1.DeMillo,R.A.andR.J.
2.Briand,L.C.
3.Zuse , H.
4Fenton,N.
5.Zuse , H.
6.Fenton,N.

جا دارد که بپرسیم، در هر حال این متریک‌های فنی تا چه میزانی معتبر هستند، به معنای آن‌که بررسی شود که تا چه اندازه متریک‌های فنی با قابلیت اطمینان بلند مدت و کیفیت سیستم کامپیوتری از نزدیک به هم پیوسته‌اند. فنتون [FEN91][1] به این سؤال به طریقهٔ ذیل پرداخته است:

علی‌رغم ارتباطات شهودی بین ساختار درونی محصولات نرم‌افزاری (متریک‌های فنی) و خصلت‌های بیرونی محصول و فرایند، عملاً کوشش‌های علمی اندکی برای برقرار کردن ارتباطات خاص، به‌عمل آمده است. چند دلیل برای این امر وجود دارد. رایج‌ترین این دلایل همانا عملی نبودن انجام آزمونهای تجربی مرتبط می‌باشد.

هر یک از "چالش‌های" فوق‌الذکر دلیلی برای احتیاط کردن است. اما این دلیلی برای صرفنظر کردن از متریک‌های فنی نیست.[2] اندازه‌گیری چنانچه خواهان کیفیت باشیم، ضروری است.

گامهای فرآیند اندازه گیری موثر کدامهاست؟

۱۹-۳-۲ اصول اندازه گیری

پیش از معرفی مجموعه‌ای از متریک‌های فنی که ۱) در ارزیابی مدل‌های تحلیلی و طراحی، کمک می‌کنند. ۲) نشانه‌ای از پیچیدگی طراحی رویه‌ای و کد منبع را فراهم ساخته و ۳) طراحی آزمون‌های مؤثر را تسهیل می کند، حائز اهمیت است که اصول اساسی اندازه‌گیری، ادراک شوند. روچه [ROC94][3] فرآیند اندازه‌گیری که خصوصیات آن به پنج گروه فعالیت بالغ می‌شود، را پیشنهاد کرده است:

هنگامی که معیارهای فنی را تعیین می کنیم، کدام قواعد و قوانین را باید مد نظر داشته باشیم؟

- **فرموله کردن**[4]. استخراج مقیاس‌ها و متریک‌های نرم‌افزاری که برای ارائه تصویری از نرم‌افزار در نظر گرفته شده، مناسب باشند.
- **گردآوری**[5]. مکانیسم به‌کار رفته برای انباشت داده‌های مورد نیاز جهت حصول متریک‌های فرموله شده.
- **تحلیل**[6]. محاسبهٔ متریک‌ها و کاربرد ابزارهای ریاضیاتی
- **تفسیر**[7]. ارزیابی نتایج متریک در تلاش برای کسب شناخت از کیفیت باز نمود.

---

1 Fenton,N.

۲. ادبیات گسترده ای از متریک های نرم افزاری بوجود آمده است (برای اطلاع عمیق به [FEN94] ، [ROC94] ، [ZUS97] مراجعه نمایید.) و نقد متریک های خاص (شامل برخی از آنها که در این فصل آمده) عمومیت یافته است – با این وجود، بسیاری از انتقاد ها متمرکز بر مسائل داخلی و اجرایی است و هدف اولیه اندازه گیری در دنیای واقعی را از یاد برده اند. کمک به مهندسین برای برقراری راهی سیستماتیک و ملموس از انجام وظائفش و پیشرفت کیفیت محصول به عنوان نتیجه نهایی.

3 Roche,J.M.

4 Formulation

5 Collection

6 Analysis

7 Interpretation

توصیه

مخصوصاً، تلاش اولیه در تعیین معیارهای فنی را نشاده انگارید. فریب متریک ها کامل را نخورید، زیرا چنین چیزی وجود ندارد

- بازخور[1]. توصیه‌های حاصل از تفسیر متریک‌های فنی منتقل شده به گروه نرم‌افزار

اصول مرتبط با فرمولاسیون متریک‌های فنی عبارت هستند از: [ROC94][2]

- اهداف اندازه‌گیری باید پیش از شروع جمع‌آوری داده‌ها، تعیین شوند.
- هر متریک فنی باید به طریقی واضح تعریف شود.
- متریک‌ها باید بر اساس نظریه‌ای که برای حوزهٔ کاربرد، معتبر است، استخراج شوند. (مثلاً، متریک‌های طراحی باید از مفاهیم و اصول طراحی اساسی استفاده کرده و برای نشان دادن حضور یک خصلت که مطلوب تلقی می‌شود، تلاش نمایند.
- متریک‌ها باید به‌نحوی آماده شوند که به بهترین حالت در خدمت محصولات و فرایندهای ویژه قرار گیرند. [BAS84][3]

هر چند فرموله‌کردن یک نقطهٔ آغاز حساس است، گردآوری و تحلیل، فعالیت‌هایی هستند که فرایند اندازه‌گیری را بدست می‌دهند. روچه [ROC94][4] اصول ذیل را برای این فعالیت‌ها پیشنهاد کرده است:

- هرگاه که ممکن باشد، جمع آوری داده‌هاو تحلیل باید به‌طور خودکار انجام شود.
- فنون آماری معتبر برای برقرارسازی ارتباط بین خصلت‌های درونی محصول و خصوصیات کیفیت خارجی (مثلاً، آیا سطح پیچیدگی معماری با تعداد نقایص گزارش شده در کاربرد محصول، هم‌خوانی دارد؟) باید به‌کار گرفته شوند.
- رهنمودهای تفسیری و توصیه‌ها باید برای هر متریک برقرار شوند.

علاوه بر اصول فوق‌الذکر، موفقیت فعالیت متریک‌ها نیز به حمایت مدیریت پیوند خورده است. تأمین بودجه، و ارتقاء و آموزش را چنانچه برنامهٔ اندازه‌گیری فنی قرار است برقرار و حفظ شود، باید در نظر گرفت.

۱۹-۲-۳ صفات خاصه متریک های موثر نرم افزار

پرسش

چگونه می توانیم کیفیت یک متریک نرم افزاری مشخص را ارزیابی نمائیم؟

صدها متریک برای نرم‌افزار کامپیوتر پیشنهاد شده‌اند ولی همهٔ آنها از مهندس نرم‌افزار حمایت عملی نمی‌کنند. بعضی نیاز به اندازه‌گیری بیش از حد پیچیده دارند و بعضی دیگر آن‌قدر ایزومتریک هستند که تعداد محدودی متخصص در دنیای واقعی امید فهم آنها را دارد و بعضی دیگر مفاهیم شهودی اساسی دربارهٔ کیفیت بالای نرم‌افزار را نقض می‌کنند.

1.Feedback
2.Roche,J.M.
3.Basili,V.R. and D.M.
4.Roche,J.M.

ای جیوگو [EJI91][1] مجموعه‌ای از خصلتها را تعریف می کند که باید توسط متریک‌های نرم‌افزاری کارآمد در برگرفته شوند. متریک حاصله و مقیاس‌های منتهی به آن باید:

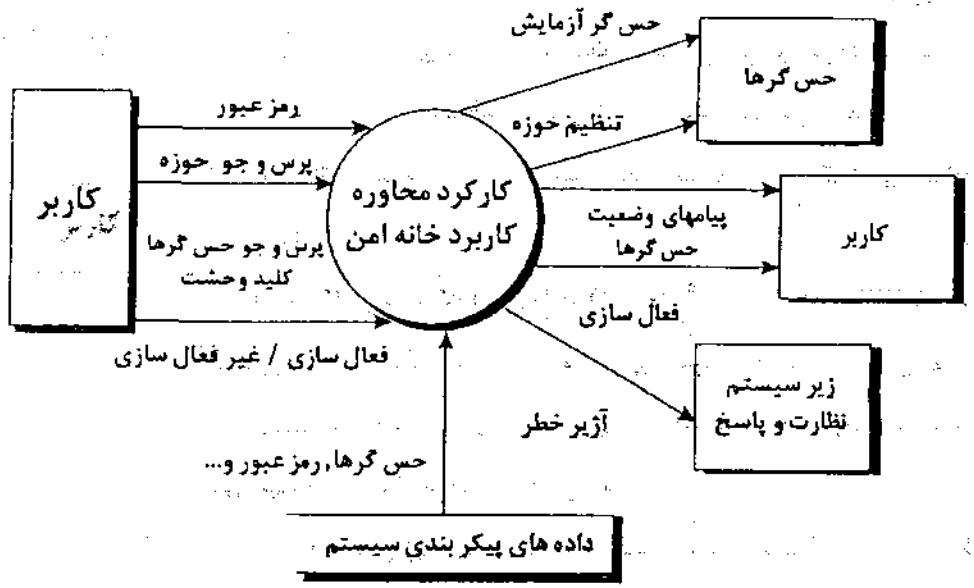
- **ساده و قابل محاسبه باشند.**[2] باید یادگیری این‌که چطور متریک استخراج می‌شود، ساده باشد و محاسبهٔ آن نباید نیاز به تلاش و زمان زیاد داشته باشد.
- **از نظر تجربی و شهودی ترغیب‌کننده باشند.**[3] متریک باید نظرات شهودی مهندس را دربارهٔ خصلت محصول تحت بررسی، ارضاء کند. (مثلاً، متریکی که چسبندگی پیمانه‌ای را اندازه‌گیری می‌کند باید از نظر مقداری همان‌طور که سطح چسبندگی افزایش می‌یابد، افزایش پیدا کند.)
- **سازگاری و دستیافتنی.**[4] متریک باید همیشه نتایجی را بدست دهد که مبهم نباشند. یک شخص ثالث مستقل باید بتواند همان مقدار متریک را با به‌کارگیری اطلاعات مشابه دربارهٔ نرم‌افزار، استخراج کند.
- **سازگاری در به‌کارگیری واحدها و ابعاد.**[5] محاسبهٔ ریاضیاتی متریک باید مقیاس‌هایی را به‌کارگیری که به ترکیبات غریب واحدها، منتهی نشود. برای مثال، افزایش تعداد افراد حاضر در تیم‌های پروژه با متغیرهای زبان برنامه‌سازی در برنامه منجر به مخلوط شدن مشکوک واحدها که به‌طور شهودی ترغیب‌کننده نمی‌باشند می‌گردد.
- **برنامه‌ریزی زبان مستقل**[6]. متریک‌ها بایید بر پایهٔ مدل تحلیل، مدل طراحی یا ساختار خود برنامه استوار باشند. آنها نباید متکی به هوس بازی‌ها و سلیقه های نحوی و ویژگیهای خاص زبان برنامه‌سازی باشند.
- **یک مکانیسم مؤثر برای بازخورد با کیفیت.**[7] هر متریک باید اطلاعات منتهی به محصولات با کیفیت بالاتر را در اختیار مهندس نرم‌افزار، قرار بدهد.

تجربه نشان می دهد که یک متریک فنی، تنها وقتی استفاده می شود که مستقیماً قابل فهم و محاسبه باشد. از اینرو متریکی که از متغیرهای متعدد و محاسبات پیچیده

هر چند اکثر متریک‌های نرم‌افزاری، خصلت‌های فوق را برآورده و ارضاء می‌کنند. با این حال، برخی از متریک‌های با کاربرد متعارف شاید در برآورده کردن یک یا دو مورد مؤفق عمل نکنند. مثالی در این مورد امتیازکارکردی (بررسی شده در فصل‌ها و دوباره در این فصل) می‌باشد. می‌توان چنین

1.Ejiogu,L.
2.Simple and computable
3.Empirically and intuitively persuasive
4.Consistent and objective
5.Consistent in its use of units and dimensions
6.Programming language independent
7.An effective mechanism for high-quality feedback

استدلال[1] کرد خصلت سازگاری و شهودی شکست می‌خورد، زیرا یک شخص ثالث مستقل شاید نتواند همان مقدار امتیاز کارکرد را به‌عنوان همکاری که همان اطلاعات را به‌کار گرفته است، استخراج کند. آیا در نتیجه باید مقیاس FP را مردود بدانیم؟ پاسخ آن است که "البته که نه!" FP شناخت خوبی را فراهم کرده و بنابراین مقدار روشنی را ارائه می‌کند. هر چند که در تأمین یک خصلت به‌طور کامل شکست می‌خورد.



شکل ۱۹-۳ بخشی از یک مدا تحلیلی برای نرم افزار خانه امن

۱۹-۳ متریک های مدل تحلیل

ارجاع به فصلها

مدل های داده ای، رفتاری و عملیاتی در فصل های ۱۱ و ۱۲ توضیح داده شده اند.

کار فنی در مهندسی نرم‌افزار با ایجاد مدل تحلیل، شروع می‌شود. در این مرحله است که نیازمندیها استخراج شده و بنیادی برای طراحی برقرار می‌گردد. بنابراین، متریک‌های فنی که نسبت به کیفیت مدل تحلیل شناخت ارائه می‌کنند، مطلوب هستند. هر چند نسبتاً تعداد محدودی متریک‌های تحلیلی و مشخص‌سازی در متون این رشته ظاهر شده‌اند. با این وصف امکان تطبیق متریک‌ها برای کاربرد پروژه (فصل ۴) جهت استفاده در این قالب، وجود دارد. این متریک‌ها مدل تحلیل را می آزمایند تا اندازهٔ سیستم حاصله را پیش‌بینی کنند. محتمل است که سایز و پیچیدگی طراحی به‌طور مستقیم با یکدیگر همبستگی داشته باشند.

۱. لطفاً توجه داشته باشید که یک آرگومان شمارشی قدرتمند برای تعادل (متریک های نرم افزار-م) می تواند ساخته شود. آنچه که در طبیعت متریک های نرم افزار وجود دارد.

۱۹-۳-۱متریک های مبتنی بر کارکرد

متریک امتیاز کارکرد (FP) در (فصل ۴) می‌تواند به‌طور کارآمدی به‌عنوان یک وسیلهٔ پیش‌بینی اندازه یک سیستم که از مدل تحلیلی حاصل خواهد شد، به‌کار برود. برای نمایش کاربرد متریک FP در این متن، ما یک مدل تحلیلی ساده را با تصاویر ارائه شده در شکل ۱۹-۳ در نظر می‌گیریم. با اشاره به شکل، یک نمودار جریان داده‌ها (فصل ۱۲) برای کارکرد در نرم‌افزار خانهٔ امن[1]، نمایش داده شده است. کارکرد مربوطه، محاوره کاربر را مدیریت کرده، اسم رمز کاربر را می‌پذیرد تا سیستم را فعال یا غیرفعال نماید و اجازهٔ کنترل وضعیت منطقه ایمن را فراهم کرده در اختیار سنسورهای متعدد ایمنی قرار می‌دهد. کارکرد یک‌سری پیام‌های ارتقاء را نمایش داده و سیگنال‌های کنترل مناسب را به اجزاء مختلف سیستم ایمنی، ارسال می‌کند. نمودار جریان داده‌ها ارزشیابی می‌شود تا مقیاس‌های کلیدی لازم برای محاسبهٔ متریک امتیاز کارکرد (فصل ۴) تعیین شوند:

- تعداد ورودی‌های کاربر
- تعداد خروجی‌های کاربر
- تعداد پرس‌و جوهای کاربر
- تعداد فایل‌ها
- تعداد رابط های خارجی

سه ورودی کاربر: اسم رمز، کلید وحشت و فعال‌سازی/ غیرفعال‌سازی در شکل مربوطه در راستای دو پرس‌و جو، نمایش داده شده‌اند: پرس و جوی منطقه ای، پرس‌وجوی سنسوری. یک فایل (فایل پیکربندی سیستم) نمایش داده شده است. دو خروجی کاربر (پیام‌ها و وضعیت سنسور) و چهار تعامل بیرونی (سنسور آزمون، تنظیم منطقه، فعال‌سازی/ غیرفعال‌سازی و آژیر خطر) نیز به تصویر درآمده‌اند. این داده‌ها همراه با پیچیدگی مناسب در شکل ۱۹-۴ نمایش داده شده‌اند.

شمارش ـ کل نمایش داده شده در شکل ۱۹-۴ باید با به‌کارگیری معادلهٔ (۱-۴) تطبیق داده شود:

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

در این معادله شمارش کل مقدار تجمعی عناصر FP حاصله از شکل ۱۹-۳ بوده و F_i ($i = ۱$ تا ۱۴) "مقادیر تطبیق پیچیدگی" هستند. در رابطه با این مثال، ما چنین تصور می‌کنیم که $\sum(F_i)$ همانا چهل و شش باشد (یک محصول نسبتاً پیچیده). بنابراین،

$$FP = 50 \times [\ 0.65 + (\ 0.01 \times 46\ )\ ] = 56$$

1.SafeHome

خانه امن، یک سیستم امنیت خانه است که به عنوان مثالی کاربردی در فصل های نخستین مورد استفاده واقع شد.

برای یک کار فنی مفید، معیارهایی که به اتخاذ تصمیمات فنی کمک می کنند، (مانند یافتن خطا طی آزمون واحد) باید گردآوری و با متریک های FP نرمالیزه شوند.

ارجاع به وب

دیباچه ای مفید بر FP توسط کپرز جونز تهیه گردیده است که در آدرس زیر قرار دارد: www.spr.com/library/ofuncmet.htm

بر پایه مقدار FP در نظر گرفته شده و حاصله از مدل تحلیلی، تیم پروژه می‌تواند سایز کل پیاده‌سازی کارکرد تعاملی کاربر خانه امن را تخمین بزند. تصور کنید که داده‌های قدیمی بیان‌گر آن است که یک FP به ۶۰ خط کد ترجمه شود (یک زبان شئ گرا باید به‌کار رود) و این‌که ۱۲ FP برای هر تلاش نفر ـ ماه تولید شده‌اند. داده‌های

گذشته، اطلاعات مهم طراحی را در اختیار مدیر پروژه قرار می‌دهد که بر پایه مدل تحلیلی استوار هستند تا محاسبات تخمین اولیه. باز هم تصور کنید که پروژه‌های قدیمی به یک میانگین سه غلط در هر امتیاز کارکرد در طول بازبینی طراحی و تحلیل رسیده‌اند و ۴ غلط در هر امتیازکارکرد در طول آزمون واحد و تمامیت این داده‌ها می‌توانند به مهندس نرم‌افزار کمک کنند تا کامل بودن بازبینی خود را ارزیابی و دست به انجام آزمون بزند.

عوامل مورد اندازه‌گیری	شمارش		ضرائب وزنی: ساده	متوسط	پیچیده		
تعداد ورودی ها	۳	×	۳	۴	۶	=	۹
تعداد خروجی ها	۲	×	۴	۵	۷	=	۸
تعداد پرس و جوهای کاربر	۲	×	۳	۴	۶	=	۶
تعداد پرونده ها	۱	×	۷	۱۰	۱۵	=	۷
تعداد واسط های خارجی	۴	×	۵	۷	۱۰	=	۲۰
شمارش کل							۵۰

شکل ۱۹-۴ محاسبه نقاط کارکرد برای کارکرد خانه امن

۱۹-۳-۲ متریک بنگ

مانند متریک امتیاز کارکردی، متریک بنگ[1] را می‌توان برای ایجاد یک بیانگر اندازه نرم‌افزاری که قرار است به‌عنوان پیامد مدل تحلیلی پیاده‌سازی شود، به‌کار برد.

متریک بنگ که توسط دیمارکو [DEM82][2] بوجود آمد، دلالتی از پیاده‌سازی مستقل از اندازه سیستم است. برای محاسبهٔ متریک بنگ، مهندس نرم‌افزار باید در درجهٔ اول یک‌سری نکات ابتدایی[3] را ـ عناصری که مربوط به مدل تحلیل بوده و در سطح تحلیل دچار تقسیمات بیشتری نخواهند شد ـ مورد

1.bang metric
2.DeMarco,T.
3.primitives

نقل قول

هنگام مواجهه با متریک های جدید، لازم است سوالاتی ابتدایی و اولیه از دیگران داشته باشیم و مهمترین آنها اینکه : " ما با این متریک ها چه کنیم ؟ " مایکل ماه و لری پوتنام

ارزیابی قرار دهد. (مقایسه‌های ریشه) امتیازات اولیه [DEM82] از طریق ارزیابی مدل تحلیل و ایجاد شمارش‌ها به شکلهای ذیل، تعیین می‌شوند:[1]

امتیازات اولیه کارکردی (FuP). تعداد تغییر شکل‌هایی (حباب‌ها) که در پایین‌ترین سطح یک نمودار جریان داده‌ها، ظاهر می‌شوند. (فصل ۱۲).

عناصر داده‌ها (DE). مقدار صفات‌خاصه یک شیء داده‌ای، عناصر داده‌ها به‌صورت ترکیبی نبوده و در فرهنگ داده‌ها ظاهر می‌شوند.

اشیاء (OB). تعداد اشیاء داده‌ای تشریح شده در فصل ۱۱.

روابط (RE). تعداد ارتباط بین اشیاء داده‌ای طبق شرح فصل ۱۲.

وضعیت‌ها (ST). تعداد وضعیت‌های قابل مشاهدهٔ کاربر در یک نمودار انتقال وضعیت (فصل ۱۲).

انتقال‌ها (TR). تعداد انتقال‌های وضعیت در نمودار انتقال وضعیت (فصل ۱۲).

علاوه بر این شش نکتهٔ اولیه (مقیاس ریشه) فوق الذکر، شمارش‌های بیشتری برای موارد ذیل تعیین می‌شوند:

امتیازات اولیه کارکردی غیر خودکار که اصلاح شده‌اند (FuPM). کارکردهایی که بیرون از محدودهٔ سیستم بوده و باید اصلاح شوند تا به سیستم جدید خدمات دهند.

عناصر داده‌های ورودی (DEI). آن عناصر داده‌هایی که برای سیستم ورودی هستند.

عناصر داده‌های خروجی (DEO). آن عناصر داده‌هایی که برای سیستم خروجی هستند.

عناصر داده‌های ذخیره شده (DER). آن عناصر داده‌ای که توسط سیستم ذخیره شده‌اند.

نشانه‌های داده‌ها (TC_i). نشانه‌های داده‌هایی (اقلام‌داده‌ای که در امتیازات‌اولیه (مقیاس‌های ریشه) کارکردی مجدداً تقسیم نمی‌شوند.) که در محدودهٔ امتیاز اولیه (مقیاس ریشه) کارکردی حضور دارند (ارزیابی شده برای هر امتیاز اولیه).

ارتباطات رابطه (RE_i). ارتباطی که شیء را در مدل داده‌ها به دیگر اشیاء متصل می‌کند.

دیمارکو (DEM 82) پیشنهاد می‌کند که اکثر نرم‌افزارها را می‌توان به یکی از دو قلمرو اختصاص داد:

کارکرد قوی[2] یا داده‌های قوی[3] بر اساس نسبت RE/FuP ، کاربردهای کارکرد قوی (که اغلب در کاربردهای مهندسی و علمی با آن مواجه می‌شویم) بر روی تغییر شکل داده‌ها تأکید می‌کند و عموماً دارای

---

۱. مخفف هایی که در پرانتز، مقابل اصول آمده است، تعداد آنها را نشان خواهد داد، برای مثال FuP مشخص می کند که اصول کارکردی در یک مدل تحلیلی چه تعداد می باشند.

2.function strong

3.data strong

ساختارهای داده‌ای پیچیده نیستند. کاربردهای داده‌های قوی (که اغلب در کاربردهای سیستم‌های اطلاعات با آنها روبه‌رو می‌شویم) تمایل به داشتن مدل‌های داده‌ای پیچیده دارند.

به معنای کاربرد کارکرد ـ قوی است $RE / FuP < 0.7$

به معنای یک کاربرد همپوشانی است $0.8 < RE / FuP < 1.4$

به معنای یک کاربرد داده‌های قوی است $RE / FuP >= 1.5$

از آن‌جا که مدل‌های مختلف تحلیل، مدل را به درجات بزرگتر یا کوچکتری از پالایش تقسیم خواهد کرد، دیمارکو توصیه می‌کند که یک میانگین شمارش نشانه برای هر امتیاز اولیه:

$$TC_{avg} = \sum TC_i / FuP$$

جهت کنترل یک شکل بودن تقسیم‌بندی در پهنهٔ مدل‌های مختلف درون یک قلمرو کاربرد، مورد استفاده واقع شوند.

برای محاسبهٔ متریک بنگ جهت کاربردهای کارکرد ـ قوی، الگوریتم ذیل به‌کار می‌رود:

set initial value of bang = 0;
do while functional primitives remain to be evaluated
Compute token-count around the boundary of primitive i
Compute corrected, FuP increment (CFuPI)
Allocate primitive to class
Assess class and note assessed weight
Multiply CFuPI by the assessed weight
bang =bang + weighted CFuPI
end do

شمارش نشانه ها (count–token) از طریق تعیین تعداد نشانه‌های جداگانه که "مشهود" هستند درون یک امتیاز اولیه، تعیین می‌شود. امکان دارد که تعداد نشانه‌ها و تعداد عناصر داده‌ها با یکدیگر تفاوت داشته باشند. چنانچه عناصر داده‌ها را بتوان از ورودی به خروجی بدون هیچ تغییر شکل درونی، حرکت داد. CFuPI اصلاح شده از یک جدول چاپ شده توسط دیمارکو، معین شده است. یک نسخهٔ بسیار خلاصه شده در ذیل آمده است:

Tc_i	CFuPI
2	1.0
5	5.8
10	16.6

15	29.3
20	43.2

وزن ارزیابی شده الگوریتم فوق‌الذکر از ۱۶ گروه مختلف امتیازات اولیه کارکردی تعریف شده توسط دیمارکو، تعیین می‌شود. وزنی با طیف ۰/۶ (مسیردهی سادهٔ داده‌ها) تا ۲/۵ (کارکردهای مدیریت داده)، بر اساس گروه نکتهٔ ابتدایی، در نظر گرفته می‌شود. برای کاربردهای داده‌ها ـ قوی، متریک بنگ با استفاده از الگوریتم ذیل، محاسبه می‌شود:

```
set initial value of bang = 0;
do while objects remain to be evaluated In the data model
    compute count of relationships for object I
    compute corrected OB increment (COBI)
    bang = bang + COBI
end do
```

COBI از جدولی که توسط دیمارکو، چاپ شده است، معین می‌گردد. یک نسخهٔ اختصاری شده در ذیل آمده است:

REi	COBI
1	1.0
3	4.0
6	9.0

زمانی‌که متریک بنگ محاسبه شده باشد، تاریخچهٔ گذشته را می‌توان برای ارتباط دادن آن با اندازه و تلاش‌لازم (نیروی انسانی مورد نیاز) به‌کار برد. دیمارکو پیشنهاد می‌دهد که سازمان نسخه‌های خود را از CFuPI ایجاد کرده و جدول COBI برای کالیبره کردن اطلاعات از پروژه‌های نرم‌افزاری کامل شده، به‌کار برد.

با اندازه گیری ویژگیهای مشخصه ها، می توان کمیت های واضحی را برای تکامل بدست آورد.

۱۹-۳-۳ متریک های کیفیت مشخصات

یویس و همکارانش [DAV93][1]، لیستی از خصوصیات قابل استفاده برای ارزیابی کیفیت مدل تحلیل و تشخیص نیازهای مربوط به آن، پیشنهاد کرده‌اند:

وضوح[2] (فقدان ابهام)، کامل بودن[1]، صحیح بودن[2]، قابل فهم بودن[3]، قابل تصدیق بودن همخوانی درونی و بیرونی[4]، قابل دستیابی بودن[5]، فشردگی[6]، قابلیت ردگیری[7]، قابلیت اصلاح شدن[8]، دقت[9] و قابل

1.Davis,A.

2.specificity

استفاده مجدد بودن[10]. به‌علاوه، نویسندگان خاطرنشان می‌کنند که مشخصات کیفیت به‌طور الکترونیکی ذخیره شده، قابل اجرا و یا حداقل قابل تفسیر بوده، توسط اهمیت و ثبات نسبی تفسیر شده، نسخه‌برداری شده، سازمان یافته بازخوانی و در سطحی درست از جزییات مشخص شده باشد.

هر چند بسیاری از خصوصیات فوق الذکر ظاهراً از نظر ماهیتی، کیفی به نظر می‌رسند، دیویس [DAV93][11] پیشنهاد می‌کند که هر یک بتوانند با به‌کارگیری یک یا چند متریک، نمایندگی شوند.[12] مثلاً ما تصور می‌کنیم که در مشخص‌سازی نظیر آن‌چه در

$$n_r = n_f + n_{nf}$$

وجود دارد، n_r نیازمندی وجود دارد و n_f تعداد نیازمندیهای کارکردی و n_{nf} تعداد نیازمندیهای غیرکارکردی (مثلاً عملکردی) است.

برای تعیین وضوح[13] نیازمندیها (فقدان ابهام)، دیویس توصیه می‌کند که متریکی بر پایه هم‌خوانی تفسیر بازبین‌های هر نیازمندی، انتخاب شود:

$$Q_1 = n_{ui} / n_r$$

که در آن n_{ui} تعداد نیازمندیهایی است که برای آنها تمام بازبین‌ها دارای تفاسیر یکسان بوده‌اند. هر چه مقدار Q به 1 نزدیک‌تر باشد، ابهام مشخص‌سازی کم‌تر خواهد بود. کامل بودن نیازهای کارکردی را می‌توان با محاسبهٔ نسبت

$$Q_2 = n_u / [n_i \times n_s]$$

تعیین کرد. در این معادله n_u تعداد نیازمندیهای کارکردی منحصر به‌فرد است، n_i تعداد ورودی‌های (محرک‌ها) تعریف شده یا القاء شده توسط مشخصات بوده و n_s تعداد وضعیت‌های مشخص شده است. نسبت Q_2 درصد کارکردهای ضروری را که برای سیستم مشخص شده‌اند، اندازه‌گیری می‌کند. در هر حال،

1.completeness
2.corretness
3.understandability
4.verifiability and external consistency
5.achievability
6.concision
7.traceability
8.modifiability
9.precision
10.reusability
11.Davis,A.
۱۲.توضیحات کلی از مشخصات متریک های کیفیتی، در حوزه بحث این فصل آمده است. برای توضیحات تفصیلی به [DAV95] مراجعه نمایید.
13.specificity

ربطی به نیازمندیهای غیر کارکردی ندارد. برای لحاظ کردن آنها در متریک کلی برای کامل بودن، ما باید میزان اعتبار نیازمندیها را در نظر بگیریم:

$$Q_3 = n_c / [n_c + n_{nv}]$$

در این معادله n_c تعداد نیازمندیهایی که به‌عنوان معتبر ارزیابی و تأیید شده و n_{nv} تعداد نیازمندیهایی که هنوز اعتبارشان مشخص نیست، خواهد بود.

نقل قول

هرآنچه اندازه گرفتنی است اندازه بگیر و هرآنچه اندازه گرفتنی نیست، به صورت اندازه گرفتنی تبدیل کن.

گالیله

۱۹-۴ متریک های مدل طراحی

غیر قابل تصور است که طراحی یک هواپیما، یک تراشه کامپیوتر جدید، یک ساختمان اداری جدید، بدون تعیین شدن اندازه‌های طراحی، تعیین متریک‌های جنبه‌های گوناگون کیفیت طراحی و به‌کارگیری آنها برای هدایت روش پیشرفت طراحی، امکان اجرا شدن، داشته باشد. و با این حساب، طراحی سیستم‌های استوار به نرم‌افزار پیچیده اغلب بدون هیچ اندازه‌گیری عملی ادامه پیدا می‌کند. جنبهٔ طعنه‌آمیز قضیه آن است که متریک‌های طراحی برای نرم‌افزار در دسترس هستند ولی اکثر قریب به‌اتفاق مهندسین هم‌چنان از وجود آنها بی‌خبر می‌مانند.

متریک‌های طراحی برای نرم‌افزار کامپیوتر، مانند دیگر متریک‌های نرم‌افزاری، دارای نقص هستند. بحث در مورد کارآیی آنها ادامه پیدا کرده است و این‌که باید به چه طریقی مورد استفاده واقع شوند. بسیاری از کارشناسان استدلال می‌کنند که آزمونهای تجربی

بیشتری قبل از آن‌که اندازه‌های طراحی را بتوان به‌کار گرفت، مورد نیاز است. و با این حال، طراحی بدون اندازه‌گیری یک آلترناتیو (گزینهٔ) غیرقابل قبول است. در بخش‌های بعدی با چند متریک طراحی رایج برای نرم‌افزار کامپیوتری را بررسی خواهیم کرد. هر یک از آنها می‌توانند شناخت بهتری را در اختیار طراح قرار داده و کمک کنند تا طراحی به سطح بالاتری از کیفیت، ارتقاء یابد.

متریک ها شفافیت ساختاری را به همراه دارند، پیچیدگی داده و سیستم با طراحی معماری مرتبط می باشند.

۱۹-۴-۱متریک های طراحی معماری

متریک‌های طراحی معماری، توجه خود را به خصوصیات معماری برنامه (فصل ۱۴) همراه با تأکید بر ساختار معماری و موثربودن پیمانه‌ها، متمرکز می‌کند. این متریک‌ها به این مفهوم که نیاز به هیچ دانشی از عملیات درونی یک جزء خاص واقع در سیستم ندارند، جعبه سیاه تلقی می‌شوند.

کارو و گلاس [CAR90][1] این سه مقیاس پیچیدگی طرح نرم‌افزاری را تعریف کرده‌اند: پیچیدگی ساختاری[2]، پیچیدگی داده‌ای[3] و پیچیدگی سیستم[4].

1.Card,D.N. and R.L.

2.Structural complexity

3.Data complexity

4.system complexity

پیچیدگی *ساختاری* یک پیمانه i به روش ذیل تعریف می‌شود:

$$S(i)=f^{2}_{Out}(i) \qquad (۱-۱۹)$$

که در آن $f_{out}(i)$ *توان خروجی*[1] پیمانه i است.

پیچیدگی *داده‌ها* مبین پیچیدگی در تعامل درونی پیمانه i می‌باشد و به صورت زیر تعریف می شود :

$$D(i) = v(i) / [f_{out}(i) + 1] \qquad (۱۹-۲)$$

که در آن v(i) تعداد متغیرهای ورودی و خروجی است که به پیمانه i وارد می شوند یا از آن خارج می گردند.

سرانجام، پیچیدگی سیستم به‌عنوان مجموع پیچیدگی ساختاری و داده‌ای بوده و به‌صورت ذیل ارائه می‌شود:

$$C(i) = S(i) + D(i) \qquad (۳-۱۹)$$

همان‌طور که هر یک از این مقادیر پیچیدگی افزایش می‌یابد. پیچیدگی معماری کل سیستم نیز افزایش پیدا می‌کند. این امر به احتمال بیشتر به افزایش یافتن میزان کار لازم برای جامعیت و آزمودن می‌انجامد.

یک متریک طراحی معماری سطح ـ بالا اولیه که توسط هنری و کافورا [HEN81][2] پیشنهاد شد، نیز از کاربرد توان ورودی، توان خروجی بهره می‌گیرد. این افراد پیچیدگی متریک شکل (قابل استفاده برای فراخوانی معماران) را به شکل زیر تعریف کردند:

$$HKM = lenqht(i) \times [f_{in}(i) + F_{out}(i)]^{2} \qquad (۴-۱۹)$$

که در آن Lenqht (طول) تعداد جملات و عبارات زبان برنامه‌نویسی در پیمانه i و $f_{in}(i)$ عبارت از توان ورودی پیمانه i، می‌باشد. هنری و کافورا تعریف ارائه شده برای توان ورودی و توان خروجی در این کتاب را نه فقط به تعداد ارتباطات پیمانه کنترل تعمیم می‌دهد، بلکه علاوه بر آن تعداد ساختارهای داده‌ای که پیمانه i از آنها داده‌ها را گرفته (Fan-in) و بیرون (Fan-out) می‌دهد. ملحوظ شده‌اند. برای

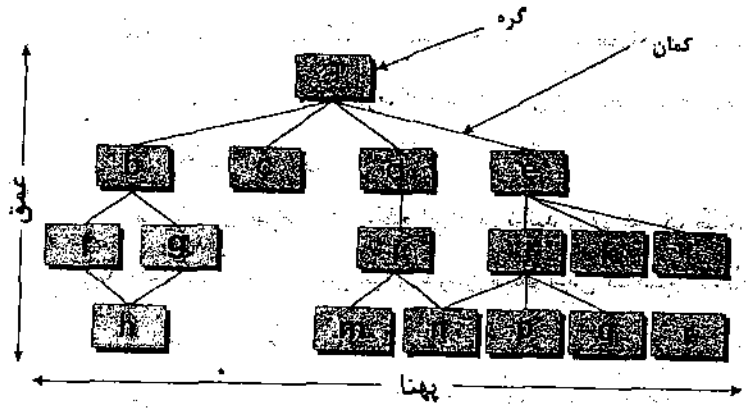
پرسش

آیا راهی برای ارزیابی میزان پیچیدگی مدل های اصلی معماری وجود دارد؟

۱. با توجه به توضیح ارائه شده فصل ۱۳، توان خروجی، تعداد زیرپیمانه های بلافصل پیمانه i را مشخص می کند. که تعداد پیمانه هایی خواهد بود که مستقیما توسط پیمانه i فراخوانده می شوند.

2.Henry,S.andD.Kafura

محاسبهٔ HKM در طول طراحی، طراحی رویه‌ای را می‌توان برای تخمین تعداد عبارات زبان برنامه نویسی برای پیمانه i، به‌کار گرفت. مانند متریک‌های کارد و گلاس که پیشتر دیدیم، افزایش متریک هنری ـ کانورا به احتمال بال، به افزایش تلاش برای جامعیت و تلفیق و آزمون یک پیمانه خواهد انجامید.



شکل ۱۹-۵ متریک های مورفولوژی

فنتون [FEN91][1] تعدادی متریک مورفولوژی[2] ساده (مانند شکل۱۹-۵) را که معماری های برنامه مختلف را قادر می‌سازد تا با استفاده از یک‌سری ابعاد مستقیم مقایسه شوند، توصیه می‌کند:

$$size = n + a$$

که در آن n تعداد گره ها و a تعداد یالها می باشد. برای معماری نشان داده شده در شکل ۱۹-۵ خواهیم داشت :

$$Size = ۱۷ + ۱۸ = ۳۵$$

عمق = طولانی‌ترین راه از گرهٔ ریشه (بالا) تا یک گرهٔ برگی. برای معماری شکل۱۹-۵ فوق عمق = ۴ است.

عرض= حداکثر تعداد گرهها در یک سطح از معماری. برای معماری نمایش داده شده در شکل ۱۹-۵ ، عرض = ۶ است.

$r = a / n$ ، نسبت یال به گره (Arc-to-node ratio,r)

که دانستیهٔ ارتباط‌دهی معماری را اندازه‌گیری کرده و می‌توان بیانی ساده از ترکیب معماری باشد.

برای معماری شکل ۱۹-۵ ،

$$r = 18 / 17 = 1.06$$

خواهد بود.

نقل قول

اندازه گیری می تواند به منزله انحراف قلمداد شود. این انحراف برخی مواقع لازم است چرا که آدمی همواره (بدون پشتیبانی کمی) امکان انتخاب و تصمیم گیری شفاف و مبتنی بر هدف را ندارد.

نرست زوسه

1.Fenton,N.

2.morphology

۱۹-۴-۲ متریک های طراحی تفصیلی (سطح اجزاء)

متریک‌های طراحی در سطح اجزاء توجه را به خصوصیات درونی اجزاء نرم‌افزار معطوف کرده و مقیاس‌های "3-CS" ـ چسبندگی پیمانه، پیوستگی و پیچیدگی را شامل می‌شود. این مقیاس‌ها می‌توانند به مهندس نرم‌افزار کمک کنند تا کیفیت طراحی در سطح اجزاء را مورد قضاوت قرار دهد. متریک‌های ارائه شده در این بخش به این مفهوم که نیاز به دانش و شناخت از عملیات درونی پیمانه تحت بررسی دارند، جعبه شیشه‌ای نامیده می‌شوند.

محاسبه میزان استقلال عملیاتی – پیوستگی و چسبندگی – یک جزء و استفاده آنها در ارزیابی کیفیت یک طراحی امکان پذیر می باشد.

متریک‌های طراحی در سطح اجزاء می‌توانند زمانی‌که یک طراحی رویه‌ای ایجاد شده باشد، به‌کار روند. بنابراین ترتیب، می‌توان آنها را تا زمان فراهم شدن کد منبع، به تأخیر انداخت.

متریک‌های چسبندگی (انسجام)، بیمن و اوت [BIE94][1] مجموعه‌ای از متریک‌ها را که نشانه‌ای از منسجم بودن و چسبندگی یک پیمانه است (فصل ۱۳) تعریف کرده‌اند:

برش داده‌ها.[2] به بیان ساده، یک برش داده‌ها نوعی قدم برداشتن رو به عقب و بررسی یک پیمانه‌ای است که در جستجوی مقادیر داده‌های تأثیرگذار به مکان پیمانه می‌باشد. باید در نظر داشت که هم برش‌های برنامه (که روی عبارات و شرایط متمرکز هستند) و هم برش‌های داده‌ها را می‌توان تعریف کرد.

نشانه‌های داده‌ها.[3] متغیرهایی که برای یک پیمانه به‌عنوان نشانه‌های داده‌ای پیمانه می‌توان تعریف کرد.

نشانه‌های چسبی.[4] مجموعه‌ای از نشانه‌های داده‌ای که در یک یا چند برش قرار می‌گیرند.

نشانه‌های سوپر ـ چسبی.[5] نشانه‌های داده‌ای که برای هر برش داده‌ای در یک پیمانه آشنا هستند.

چسبی بودن.[6] چسبی بودن نسبی، مستقیماً با تعداد برش‌های داده‌ای که به هم پیوند می‌زنند، متناسب است.

بیمن و اوت متریک‌هایی برای انسجام و چسبندگی کارکردی قوی[7] (SFC)، انسجام و چسبندگی کارکردی ضعیف[1] (WFC) و توان چسبندگی[2] (میزان نسبی چسباندن برش‌های داده‌ای به یکدیگر توسط نشانه‌های چسبی). این متریک‌ها را به‌طریق ذیل می‌توان تفسیر کرد:

1.Bieman,J.M. and L.M.

2.Data slice

3.Data tokens

4.Glue tokens

5.*Superglue tokens*

6.Stickiness

7.Strong Functional Cohesion

تمام این متریک‌های انسجام از نظر مقداری دارای طیفی صفر و یک هستند. آنها دارای مقدار صفر هستند، زمانی‌که یک رویه دارای بیش از یک خروجی باشد و هیچ‌یک از خصلت‌های انسجام بیان شده توسط یک متریک خاص را نمایش ندهد.

یک رویه فاقد نشانه‌های سوپر - چسبی و یا نشانه‌ای که برای تمام برش‌های داده‌ای آشنا و مشترک باشد، دارای انسجام کارکردی قوی صفر است، به این معنا که دارای هیچ نشانهٔ چسبی نبوده - یعنی هیچ نشانهٔ مشترک با بیش از یک برش داده‌ای نداشته باشد (در رویه‌هایی با بیش از یک برش داده‌ای)، نمایشگر انسجام کارکردی ضعیف با مقدار صفر است و توان چسبندگی نیز صفر است. هیچ نشانه داده‌ای که در خدمت بیش از یک خروجی باشد، وجود ندارد.

با انسجام کارکردی قوی و توان چسبندگی زمانی روبرو می‌شویم که بیمن و اوت یک مقدار حداکثری را به‌کار می‌برند، روبه‌رو می‌شویم.

بحث تفصیلی دربارهٔ متریک‌های بیمن و اوت بهتر است برای نویسندگان باقی بماند. برای نمایش کاراکتر این متریک‌ها، متریک برای انسجام کارکردی قوی را در نظر بگیرید:

$$SFC(i)=SG\,[SA(i)]\,/\,[tokens(i)] \qquad (۶-۱۹)$$

که در آن SG[SA(i)] نشانگر نشانه‌های سوپر - چسبی است - سری نشانه‌های داده‌ای که روی تمام برش‌های داده‌ای یک پیمانه i قرار می‌گیرند. همان‌طور که نسبت نشانه‌های سوپر - چسبی به تعداد کل نشانه‌ها در پیمانه i به سمت حداکثر ارزش و مقدار 1 نزدیک شده و افزایش می‌یابد، انسجام کارکردی پیمانه نیز افزایش می‌یابد.

متریک‌های پیوستگی (متصل‌کننده). اتصال پیمانه تعاملی از "متصل بودن" یک پیمانه با دیگر پیمانه‌ها، داده‌های سراسری و محیط خارجی، ارائه می‌کند. در فصل ۱۳، متصل‌سازی از لحاظ کیفی بررسی شد.

داهما [DHA95][3] متریکی را برای متصل کردن پیمانه پیشنهاد کرده است که در بردارندهٔ جریان داده‌ها و کنترل پیوستگی و متصل‌سازی، متصل‌سازی سراسری و متصل‌سازی محلی است. مقیاس‌های مورد نیاز برای محاسبهٔ پیمانه متصل‌کننده از لحاظ هر یک از سه نوع متصل‌سازی فوق‌الذکر تعریف شده‌اند.

برای متصل‌سازی جریان داده‌ها و کنترل،[4]

d_i = تعداد پارامترهای ورودی داده‌ها

ارجاع به وب

مقاله ای با عنوان " یک سیستم متریک نرم افزار برای پیوستگی پیمانه" قابل پیاده سازی از آدرس زیر می باشد :

www.isse.gmu.edu/faculty/ofut/rsrch/abstracts/mi-coupling.html

1.weak functional cohesion

2.adhesiveness

3.Dhama,H.

4. For data and control flow coupling

c_i = تعداد پارامترهای ورودی کنترل

c_o = تعداد پارامترهای خروجی داده‌ها

c_o = تعداد پارامترهای خروجی کنترل

برای متصل‌سازی جهانی،[1]

g_d = تعداد متغیرهای سراسری به‌کار رفته به‌عنوان داده‌ها

g_c = تعداد متغیرهای سراسری به‌کار رفته به‌عنوان کنترل

برای متصل‌سازی محیطی،[2]

w = تعداد پیمانه‌های خوانده شده (توان خروجی)

r = تعداد پیمانه‌هایی که پیمانه‌های تحت بررسی را فراخوان می‌کنند. (توان ورودی)

با به‌کارگیری این مقیاس‌ها، یک نمایش‌گر متصل‌سازی پیمانه، m_c، به روش ذیل تعریف شده است:

$$m_c = K / M$$

که در آن K=1 و یک ثابت تناسبی[3] است.

$$M = d_i + (a \times c_i) + d_o + (b \times c_o) + g_d + (c \times g_c) + w + r$$

در آن a=b=c=2 می‌باشد.

هر چه مقدار m بیشتر باشد. توان اتصال کردن کلی پیمانه پایین‌تر است. مثلاً، اگر پیمانه‌ای یک پارامتر ورودی و خروجی داده‌های منفرد داشته باشد. به هیچ دادهٔ سراسری دست نیافته و یک پیمانه منفرد خوانده می‌شود:

$$m_c = 1 / (1 + 0 + 1 + 0 + 0 + 0 + 1 + 0) = 1/3 = 0.33$$

ما انتظار داشتیم که چنین پیمانه ای اتصال کمتری انجام دهد. از این بابت، یک مقدار m_c=0.33 نشان‌گر اتصال‌دهی کم است. به‌همین ترتیب، اگر پیمانه ای دارای ۵ پارامتر ورودی و ۵ پارامترخروجی داده‌ها باشد، تعداد برابری پارامترهای کنترل، به دسترسی بر ۱۰ قلم داده‌های سراسری، و دارای توان ورودی ۳ و توان خروجی ۴ می‌باشد.

$$mc = 1 / [5 + (2 * 5) + 5 + (2 * 5) + 10 + 0 + 3 + 4] = 0.02$$

و توان اتصال‌دهی بالا خواهد بود.

1.For global coupling

2.For environmental coupling

۳. نویسنده [DHA95] تذکر می دهد که مقادیر k و a و b و c (که در معادله بعد توضیح داده شده) می توانند با تجربیات بیشتر از رویه های تعیین صحت، تنظیم شوند.

برای آن‌که متریک اتصال‌دهی همان‌طور که میزان اتصال‌دهی افزایش می‌یابد، به طرف بالا حرکت کند (یک خصلت مهم که در بخش ۱۸-۲-۳ مورد بررسی واقع شده است)، یک متریک اتصال‌دهی، تجدیدنظر شده را می‌توان چنین تعریف کرد:

$$C = 1 - m_c$$

پیچیدگی سیکلوماتیک، تنها یکی از انواع متنوع پیچیدگی می باشد.

که در آن میزان اتصال‌دهی به‌صورت غیر خطی مابین یک مقدار حداقلی در دامنهٔ ۰/۶۶ تا مقدار حداکثری ۱/۰ افزایش پیدا می‌کند.

متریک‌های پیچیدگی. انواع متریک‌های نرم‌افزاری را می‌توان برای تعیین پیچیدگی جریان کنترل برنامه، محاسبه کرد. بسیاری از اینها بر پایه گراف جریان استوار شده‌اند. همان‌طور که در فصل ۱۷ بررسی شد، یک گراف از گره‌ها و پیوندها تشکیل یافته که اتصال‌ها و لبه‌ها[1] نیز نامیده می‌شوند. زمانی‌که لبه‌ها جهت‌دار باشند، گراف جریان یک گراف جهت‌دار[2] خواهد بود.

مک کیب [Mcc94][3] تعدادی از کاربردهای مهم متریک‌های پیچیدگی را تعریف کرده است:

متریک‌های پیچیدگی را می‌توان برای پیش‌بینی اطلاعات حساس دربارهٔ قابلیت اطمینان و قابلیت نگهداری سیستم‌های نرم‌افزاری از یک کد منبع تحلیل خودکار، به‌کار گرفت [یا اطلاعات طراحی رویه‌ای] متریک‌های پیچیدگی علاوه بر آن، در طول پروژه نرم‌افزار فیدبک (باز خورد) فراهم می‌کنند تا به کنترل [فعالیت طراحی] کمک نمایند. در طول انجام آزمون و نگهداری، آنها اطلاعات تفصیلی دربارهٔ پیمانه‌های نرم‌افزاری فراهم می‌کنند که به تشخیص و کشف محدوده‌های بی‌ثباتی بالقوه کمک می‌کند.

یکی از پر کاربردترین (و بحث‌انگیزترین) متریک‌های پیچیدگی برای نرم‌افزار کامپیوتری پیچیدگی سیکلوماتیک که در اصل توسط توماس مک کیب [MCC76][4] و [MCC89][5] ایجاد شده و به‌طور مفصل در بخش ۱۷-۴-۲ بررسی شده است.

متریک مک کیب یک مقیاس کمیتی برای آزمون میزان مشکل بودن آزمون و مبین حداکثر قابلیت اطمینان می‌باشد. مطالعات تجربی روابط مشخصی بین متریک مک کیب، تعداد خطاهای موجود در برنامه منبع و نیز زمان مورد نیاز برای یافتن و تصحیح این‌گونه اشتباهات را نشان می‌دهند.

مک کیب هم‌چنین معتقد است که پیچیدگی چرخشی یا سیکلوماتیک را بتوان برای ارائه مبین کمیتی از حداکثر اندازه پیمانه، به‌کار گرفت. با جمع‌آوری داده‌ها از تعدادی پروژه‌های برنامه‌سازی واقعی، او دریافت که پیچیدگی سیکلی = ۱۰ یک حد بالایی عملی برای سایر پیمانه به‌نظر می‌رسد. زمانی‌که

1.edges

2.directed

3.McCabe,T.J.and Watson

4.McCabe,T.J.

5.McCabe,T.J. and C.W. Butler

پیچیدگی چرخشی پیمانه‌ها از این عدد (تعداد) فراتر رود، دیگر آزمون یک پیمانه، بسیار سخت و مشکل می‌شود. به فصل ۱۷ در مورد پیچیدگی سیکلی به‌عنوان راهنمایی برای طراحی موارد آزمون جعبهٔ سفید رجوع کنید.

نقل قول

زیبایی، تعادل و تنظیم تمام قسمتها به تناسب است. بنابرین کسی نمی‌تواند بدون آسیب رساندن به هارمونی کل، چیزی را اضافه یا کم کند یا تغییر دهد.

لئون آلبرتی

۱۹-۴-۳ متریک های طراحی رابط کاربر

هر چند نوشته‌های زیادی دربارهٔ طراحی تعاملات انسان ـ کامپیوتر (فصل ۱۵) وجود دارد، نسبتاً اطلاعات اندکی دربارهٔ متریک‌های شناخت از کیفیت و قابلیت استفاده رابط، در دسترس است.

سیرز [SEA93][1] توصیه می‌کند که مناسب بودن طرح‌بندی اولیه[2] (LA) می‌تواند یک متریک طراحی حائز ارزش برای تعاملات انسان ـ کامپیوتر باشد. یک GUI متعارف از عناصر موجودیتهای طرح‌بندی استفاده می‌کند ـ نمادهای گرافیکی، متن، منوها، پنجره‌ها و موارد مشابه ـ تا به کاربر در تکمیل و انجام وظائف کمک کند. برای انجام کاری با استفاده از GUI، کاربر باید از یک عنصر طرح‌بندی به عنصر بعدی در حرکت باشد. موقعیت مطلق و نسبی هر عنصر طرح‌بندی، موارد و تعدد به‌کارگیری و "هزینهٔ" انتقال از یک عنصر طرح کلی به بعدی همگی در خدمت مناسب بودن تعامل قرار می‌گیرند. برای هر یک طرح‌بندی مشخص (به‌عبارت دیگر یک طراحی GUI مشخص) می‌توان هزینه را برای هر توالی اقدامات مطابق با رابطهٔ ذیل در نظر گرفت:

$$\text{هزینه} = \Sigma \left[(\text{تعداد انتقال (k)}) \times (\text{هزینه انتقال (k)}) \right] \quad (۱۹-۷)$$

توصیه

هرچند متریک های طراحی رابط عالی اند، اما از آن بالاتر حصول اطمینان از این امر است که کاربران نهایی شما، رابط را دوست دارند و در محاوره با آن از راحتی و آرامش برخوردار می باشند.

که در آن k یک انتقال مشخص از یک موجودیت طرح‌بندی به بعدی به‌عنوان یک وظیفهٔ مشخص انجام شده، می‌باشد. جمع‌بندی در تمامی انتقال‌ها برای یک کار مشخص یا سری وظائف مورد نیاز جهت انجام برخی کارکردهای به‌کارگیری رخ می‌دهد. هزینه را می‌توان از لحاظ زمان، تأخیر در پردازش، یک یا چند مقدار معقول مانند فاصله‌ای‌که ماوس باید بین موجودیتهای طرح‌بندی حرکت کند، تعیین نمود. مناسب بودن طرح کلی به طریقهٔ ذیل تعریف شده است:

$$LA = [(\text{هزینهٔ طرح بندی پیشنهاد شده}) / (\text{طرح‌بندی بهینه شده - هزینهٔ LA})] \quad (۱۹-۸)$$

که در آن LA برای طرح‌بندی بهینه، برابر با ۱۰۰ خواهد بود.

برای محاسبه طرح‌بندی بهینه یک رابط کاربر گرافیکی وضعیت واقعی رابط (ناحیه صفحه نمایش) به صورت یک توری یا یک شبکه درخواهد آمد. هر بخش از این صفحه قسمت بندی شده، یک موجودیت را بازنمایی خواهد نمود. بنابرین برای یک توری (بخش بندی صفحه نمایش برای طراحی این رابط به توری تشبیه شده است م.) با N موقعیت ممکن، K طرح بندی مختلف برای جای دادن موجودیت ها خواهیم داشت. تعداد طرح بندی ممکنه به قرار ذیل می باشد : [SEA93]

1.Sears,A.

2.Layout Appropriateness

تعداد طرح بندی ممکن = $[N! / (K! \times (N-K)!)] \times K!$ (۹-۱۹)

همان‌طور که موقعیت‌های طرح‌بندی افزایش می‌یابد، تعداد طرح‌بندی‌ها به مقدار بسیار زیادی رشد پیدا می‌کند. برای یافتن طرح‌بندی بهینه (کمترین هزینه) سیرز [SEA93][1] یک الگوریتم درختی جستجو را توصیه کرده است.

برای ارزیابی انواع پیشنهادات گوناگون طرح‌بندی GUI و حساسیت یک طرح‌بندی خاص به تغییرات در شرح‌های وظیفه موردنظر (به‌عبارت دیگر تغییرات در توالی‌ها یا تعداد انتقال‌ها)، LA به‌کار خواهد رفت. طراح رابط، می‌تواند تغییر به‌وجود آمده در مناسب بودن طرح‌بندی، ΔLA ، را به‌عنوان راهنمای انتخاب بهترین طرح‌بندی GUI جهت یک کاربرد خاص، را مورد استفاده قرار دهد.

حائز اهمیت است که دقت داشته باشیم، انتخاب یک طراحی GUI می‌تواند توسط متریک‌هایی مانند LA، هدایت شود، اما قضاوت‌کنندهٔ نهایی باید ورودی کاربر بر پایه نمونه‌های اولیه GUI باشد. نیلسن و لوی [NIE94][2] گزارش می‌دهند که "... هر فردی دارای شانس به‌طور معقول زیادی برای موفقیت است چنانچه مابین انواع طراحی ها، طراحی بر پایه صرفاً نظرات کاربر را، انتخاب کند. میانگین عملکرد وظیفه کاربر و برآورده شدن نیاز آنها توسط یک GUI، شدیداً" با یکدیگر پیوستگی دارند.

نقل قول

مغز آدمی قواعد جدی تری را نسبت به الگوریتمهای ساخته شده(بشری) دنبال می کند. موریس هالستید

۱۹-۵ متریک های برنامه منبع

تئوری هالستید در زمینهٔ علم نرم‌افزار [HAL77][3] احتمالاً بسیار شناخته شده و کاملاً مورد بررسی واقع شده است ... که مقیاس‌های ترکیبی برای پیچیدگی (نرم‌افزار) ارائه داده است. [CUR80][4] ، علم نرم‌افزار[5] اولین سری از "قوانین" تحلیلی نرم‌افزار کامپیوتری[6] را در نظر گرفته است.

علم نرم‌افزار قوانین کمیتی را برای توسعهٔ نرم‌افزار کامپیوتری در نظر گرفته و یکسری مقیاس‌های اولیه که می‌توان پس از ارائه برنامه بدان‌ها دست یافت و یا حدوداً زمانی که طراحی کامل شده است، را به‌کار می‌گیرد که در ذیل آمده‌اند:

n_1= تعداد عملگرهای مشخص که در برنامه ظاهر می‌شوند.

n_2= تعداد عملوندهای مشخص که در برنامه ظاهر می‌شوند.

1.Sears,A.

2.Nielsen,J.

3.Hastead,M.

4.Curtis,W.

5.Software science

۶. قابل ذکر است که قوانین هالستید، بحث های اساسی وجدی ایجاد نموده است و کسی معتقد نیست که نظریه بدون اشکال می باشد. با این وجود، تعیین صحت تجربی یافته های هالستید، برای تعدادی از زبانهای برنامه سازی به کار رفت. [FEL89]

N_1= تعداد کل موارد وقوع عملگر

N_2= تعداد کل موارد وقوع عملوند

عملگرها مشتملند بر تمام جریان ساختارهای کنترل، شرایط، و عملگرهای ریاضی. عملوندها تمام متغیرهای برنامه و ثابت ها را شامل می باشند.

هالستید از مقیاس‌های اولیه برای ایجاد بیان‌گرهایی جهت طول کل برنامه[1]، حداقل حجم[2] بالقوهٔ یک الگوریتم، حجم واقعی[3] (تعداد بیت‌های مورد نیاز برای مشخص‌سازی یک برنامه) سطح برنامه[4] (مقیاس برای پیچیدگی نرم‌افزار)، سطح زبان[5] (برنامه‌نویسی) ، (یک مقدار ثابت برای هر زبان) و دیگر ویژگی‌ها مانند تلاش برای توسعه، زمان توسعه، و حتی تعداد خطاهای پیش‌بینی شده در نرم‌افزار، به‌کار می گیرد.

هالستید نشان می‌دهد که طول N را می‌توان تخمین می‌زد:

$$N = n_1 \log_2 n_1 + n_2 \log_2 n_2 \qquad (۱۰-۱۹)$$

و حجم برنامه را می‌توان این‌گونه تعریف کرد:

$$V = N \log_2 (n_1+n_2) \qquad (۱۱-۱۹)$$

باید دقت کرد که V با هر زبان برنامه‌نویسی، تغییر کرده و نمایندهٔ حجم اطلاعات (به‌صورت بیت) مورد نیاز برای مشخص‌سازی یک برنامه است.

به‌طور تئوریک، یک حداقل حجم باید برای هر الگوریتم خاص، وجود داشته باشد. هالستد یک نسبت حجم فشرده‌ترین حالت یک برنامه یا حجم برنامهٔ واقعی، را تعریف می‌کند. در عمل، L باید همیشه کمتر از یک باشد. از لحاظ مقیاس‌های اولیه، نسبت حجم می‌تواند به‌طریق ذیل ارائه شود:

$$L = 2 / n_1 \times n_2 / N_2 \qquad (۱۲-۱۹)$$

کار هالستید با تصدیق و تأیید تجربی سازگار بوده و تحقیقات گسترده‌ای در زمینهٔ علم نرم‌افزار، انجام شده است. بحث دربارهٔ این کار فراتر از حوزهٔ عمل این کتاب است، اما می‌توان گفت که توافق خوبی مابین نتایج پیش‌بینی شده به‌صورت تحلیلی و نتایج تجربی، وجود دارد. برای اطلاعات بیشتر به [ZUS90][6]، [FEN91][7] و [ZUS94][8] مراجعه کنید.

1.program length

2.potential minimum volume

3.actual volume

4.program level

5.language level

6.Zuse,H.

7.Fenton,N.

8.Zuse,H.

۱۹-۶ متریک های آزمون

هر چند مطالب بسیاری دربارهٔ متریک‌های نرم‌افزاری برای انجام آزمون، نوشته شده است [HET93]، با این حال اکثر متریک‌های پیشنهاد شده بر روی فرآیند آزمون متمرکز شده‌اند و به خصوصیات خود آزمون‌ها توجه نکرده‌اند. در کل، آزمون‌کننده‌ها، باید بر روی تحلیل، طراحی، و متریک‌های برنامه جهت هدایت به‌سوی طراحی و انجام موارد آزمون، اتکاء داشته باشند. متریک‌های استوار بر کارکرد (بخش ۱۹-۳-۱) را می‌توان به‌عنوان پیش‌بینی‌کننده کل تلاش مربوط به انجام آزمون، مورد استفاده قرار داد. انواع گوناگون خصوصیات سطح پروژه (مثلاً تلاش و زمان آزمون، اشتباهات آشکار شده، تعداد مورد آزمون انجام شده) برای پروژه‌های پیشین را می‌توان گردآوری و با تعداد FP به‌وجود آمده توسط یک تیم پروژه به یکدیگر پیوند داد. آن‌گاه این تیم می‌تواند "مقادیر مورد انتظار" برای این خصوصیات پروژهٔ جاری را به کاربرد.

متریک بنگ می‌تواند دلالتی برای تعداد موارد آزمون مورد نیاز از طریق آزمون مقیاس‌های اولیه بحث شده در بخش ۱۹-۳-۲ ارائه کند. تعداد امتیازات اولیه کارکردی (FuP)، عناصر داده‌ها (DE)، اشیاء (OB)، روابط (RE)، وضعیت‌ها (ST) و انتقال‌ها (TR) را می‌توان برای پروژه‌ای کردن انواع آزمون‌های جعبه سفید و جعبه سفید نرم‌افزار به‌کار گرفت.

متریک های آزمون در دو گروه گسترده قرار می گیرند: (۱) متریک هایی که پیش بینی کننده اموری چون تعداد آزمون های مورد نیاز در سطوح مختلف می باشند و (۲) متریک هایی که بر پوشش آزمون برای یک جزء مشخص تمرکز دارند.

مثلاً، تعداد آزمون‌های مرتبط با یک تعامل انسانی را می‌توان بدین روش‌ها تخمین زد: ۱) آزمون تعداد انتقال‌ها (TR) که در باز نمود انتقال وضعیت HCI و ارزیابی آزمون‌های مورد نیاز برای هر انتقال لحاظ شده‌اند. ۲) آزمون تعداد اشیاء داده‌ای (OB) که در سرتاسر تعامل حرکت می‌کنند و ۳) تعداد عناصر داده‌ها که ورودی یا خروجی هستند.

متریک‌های طراحی معماری، اطلاعاتی در مورد راحتی یا سختی مرتبط با آزمون تمامیت و جامعیت (فصل ۸) ارائه کرده و نیاز به آزمون تخصصی نرم‌افزار (مثلاً مخرک ها) را می‌یابد. پیچیدگی سیکلوماتیک (یک متریک طراحی سطح جزء) در هستهٔ آزمون مسیر پایه، ریشه دارد. یک روش طراحی آزمون موردی که در فصل ۱۷ ارائه شده است. علاوه بر آن، پیچیدگی سیکلوماتیک را می‌توان برای پیمانه‌های هدف به‌عنوان کاندیداهای آزمون واحد گسترش‌یافته (فصل ۱۸) به‌کار گرفت. پیمانه‌های با پیچیدگی سیکلوماتیک بالا، امکان ایجاد خطای بیشتری را نسبت به پیمانه‌های با پیچیدگی کمتر دارند. به‌همین دلیل، کسی که آزمون را انجام می‌دهد باید تلاش بیشتری برای آشکار کردن اشتباهات در پیمانه پیش از آن‌که به‌صورت یکپارچه با سیستم درآید، مبذول دارد. تلاش انجام آزمون هم‌چنین می‌تواند با استفاده از متریک‌های حاصل از مقیاس‌های هالستید (بخش ۱۹-۵)، تخمین زد. با استفاده از تعاریف مربوط به حجم برنامه "V" و سطح برنامه PL، تلاش علم نرم‌افزار، e را بدین طریق می‌توان محاسبه کرد:

$$PL = 1 / [(n_1 / 2) * (N_2 / n_2)] \qquad (۱۹-۱۳ \text{ الف})$$

$$e = V / PL \qquad (۱۹-۱۳ \text{ب})$$

در صد تلاش کلی برای آزمون، که باید به پیمانه k اختصاص پیدا کند را می‌توان با به‌کارگیری رابطه ذیل، تخمین زد:

$$\text{درصد نیروی لازم برای آزمون پیمانه } (k) = e(k) / \sum e(i) \qquad (۱۹-۱۴)$$

که در آن e(k) برای پیمانه K با استفاده از معادلات (۱۹-۱۳) می‌تواند محاسبه و تخمین زده شود و جمع‌بندی در مخرج کسر معادله (۱۹-۱۴) مجموع تلاش علم نرم‌افزار در سرتاسر کل پیمانه‌های سیستم است.

همان‌طور که آزمون‌ها انجام می‌شوند، به مقیاس متفاوت بر کامل شدن انجام آزمون دلالت می‌کنند. یک مقیاس برنامه آزمون، نشانه‌ای از آن است که چه تعداد نیاز (از کل تعداد نیازها) مورد آزمون قرار گرفته‌اند. این امر نشانه‌ای از کامل شدن طرح آزمون است. عمق آزمون مقیاسی از درصد مسیرهای پایه مستقل تحت پوشش آزمون در قبال تعداد کل مسیرهای پایه در برنامه است. یک تخمین دقیق به‌طور معقول از تعداد مسیرهای پایه را می‌توان توسط افزودن پیچیدگی سیکلوماتیک تمام پیمانه‌های برنامه، انجام و محاسبه شود. سرانجام، همان‌طور که آزمون‌ها انجام می‌شوند و داده‌های خطا جمع‌آوری می‌شوند، پروفایل‌های خطا را می‌توان برای اولویت‌بندی و گروه‌بندی خطاهای آشکار شده، به‌کار گرفت. اولویت، بیان‌گر شدت مشکل است. گروه‌بندی‌های خطا، شرحی از یک خطا را ارائه می‌دهند تا بتوان تحلیل خطای آماری انجام داد.

۱۹-۷ متریک های نگهداری

تمام متریک‌های نرم‌افزار معرفی شده در این فصل را می‌توان برای توسعهٔ نرم‌افزاری جدید و نگهداری نرم‌افزار موجود، به‌کار گرفت. به‌هرحال، متریک‌های طراحی شده به‌طور صریح جهت نگهداری، پیشنهاد شده‌اند.

IEEE Std. 982.1-1988 - [IEE94] یک نمایهٔ پختگی (بلوغ) نرم‌افزار[2] (SMI) را توصیه می‌کند که دلالتی بر ثبات یک محصول نرم‌افزاری (بر پایه تغییراتی که برای هر عرضه محصولی رخ می‌دهند) ارائه می‌دهد. اطلاعات ذیل معین شده‌اند:

M_T= تعداد پیمانه‌های تولید جاری

F_C= تعداد پیمانه‌های تولید جاری که تغییر کرده‌اند.

F_a= تعداد پیمانه‌های تولید جاری که اضافه شده‌اند.

1.Hetzel,B.

2.Software Maturity Index

F_d= تعداد پیمانه‌های ناشی از تولید قبلی که در تولید فعلی حذف شده‌اند.

شاخص بلوغ نرم‌افزار به روش زیر محاسبه می‌شود:

$$SMI = [M_T - (F_a + F_c + F_d)] / M_T \qquad (۱۵-۱۹)$$

همان‌طور که SMI به طرف ۱/۰ نزدیک می‌شود، محصول شروع به ثبات یافتن می‌کند. SMI را هم‌چنین می‌توان به‌عنوان متریکی برای طراحی امور نگهداری نرم‌افزار به‌کار گرفت. میانگین زمان عرضهٔ یک محصول به‌عنوان یک محصول نرم‌افزاری را می‌توان به SMI پیوست داده و مدل‌های تجربی برای تلاش نگهداری را می‌توان توسعه داد.

۱۹-۸ خلاصه

متریک‌های نرم‌افزاری یک روش مقداری برای ارزیابی کیفیت خصلت‌های درونی محصول ارائه کرده و بدین طریق مهندس نرم‌افزار را قادر می‌سازد تا ارزیابی کیفیت را قبل از تولید محصول، انجام دهد. متریک‌ها شناخت لازم برای ایجاد مدل های موثر تحلیل و طراحی، برنامه مستحکم و ، آزمون‌های جامع را فراهم می‌کنند.

یک متریک نرم‌افزاری، برای آن‌که در دنیای واقعی مفید باشد، باید ساده و قابل محاسبه باشد و نیز ترغیب‌کننده برای استفاده، سازگار و البته ملموس و فهمیدنی باشد. یک متریک باید مستقل از زبان برنامه‌نویسی بوده و بازخورد مؤثری برای مهندس نرم‌افزار، فراهم کند. متریک‌های مربوط به مدل تحلیل روی سه جزء مدل تحلیلی یعنی کارکرد، داده‌ها و رفتار، متمرکز هستند. امتیازکارکرد و متریک بنگ هر کدام یک وسیلهٔ کمی برای ارزیابی مدل تحلیلی، فراهم می‌سازد. متریک‌های مربوط به طراحی، مقوله‌های معماری، طراحی سطح اجزاء و طراحی رابط را در نظر می‌گیرند. متریک‌های طراحی معماری جنبه‌های ساختاری یک مدل طراحی را در نظر می‌گیرند. متریک‌های طراحی سطح اجزاء، دلالتی بر کیفیت پیمانه از طریق برقراری مقیاس‌های غیرمستقیم برای انسجام، اتصال‌دهی و پیچیدگی، ارائه می‌کنند. متریک‌های طراحی رابط، دلالتی بر مناسب بودن طرح‌بندی برای GUI فراهم می‌کنند.

علم نرم‌افزار مجموعه‌ای هیجان‌انگیز از متریک‌ها را در سطح برنامه منبع فراهم می‌کند. با به‌کارگیری تعداد عملگرها و عملوندهای حاضر در کد برنامه، علم نرم‌افزار انواع گوناگون متریک‌ها که برای ارزیابی کیفیت برنامه به‌کار می‌روند، فراهم می‌کند.

تعداد اندکی متریک‌های فنی برای کاربرد مستقیم در آزمون نرم‌افزار و نگهداری، توصیه و پیشنهاد شده‌اند. به‌هر حال، بسیاری دیگر از متریک‌های فنی که می‌توانند برای راهنمایی فرآیند آزمون استفاده شوند و نیز مکانیسمی برای ارزیابی قابلیت نگهداری برای کامپیوتر ایجاد نمایند، نیز وجود دارند.

مسایل و نکاتی برای تفکر و تعمق بیشتر

۱۹-۱ نظریه اندازه‌گیری عنوانی پیشرفته می باشد که تأثیری قوی بر متریک های نرم‌افزاری داشته است. با استفاده از [ZUS97]، [FEN91] ، [ZUS97] ، [KYB84] یا منابع دیگر، مقاله ای کوتاه بنویسید و دغدغه های اصلی نظریهٔ اندازه‌گیری را طرح نمایید. پروژه انفرادی: ارائه ای در خصوص این موضوع در کلاس داشته باشید.

۱۹-۲ عوامل کیفیت مک‌کال در دهه ۱۹۷۰ توسعه یافت. تقریباً تمام جنبه‌های کامپیوتر، از آن زمان که ساخته شد، دگرگون شده‌اند ولی عوامل یاد شده هم اکنون نیز قابل استفاده اند. از این حقیقت به چه نتیجه ای می رسید؟

۱۹-۳ چرا نمی‌توان یک متریک منفرد و جامع برای پیچیدگی برنامه یا کیفیت برنامه تعیین نمود؟

۱۹-۴ مدل تحلیلی که به عنوان بخشی از مسئله ۱۲ـ۱۳ توسعه دادید، مرور نمایید. با استفاده از رهنمودهای ارائه شده در بخش ۱۹-۳-۱، تعداد امتیازیات کارکردی مرتبط با PHTRS را برآورد کنید.

۱۹-۵ مدل تحلیلی که به عنوان بخشی از مسئله ۱۲ـ۱۳ توسعه دادید، مرور نمایید. با استفاده از رهنمودهای ارائه شده در بخش ۱۹-۳-۲، شمارش اولیه را برای متریک Bang به‌دست آورید. سیستم PHTRS از لحاظ عملیات قوی است یا از لحاظ داده‌ها؟

۱۹-۶ مقدار متریک Bang را با استفاده از اندازه هایی که در مسئله ۱۹ـ۵ توسعه دادید، محاسبه کنید.

۱۹-۷ یک مدل کامل طراحی برای سیستمی که استادتان پیشنهاد می نماید، ایجاد کنید. پیچیدگی ساختاری و داده‌ای را با استفاده از متریک های شرح داده شده در بخش ۱۹-۴-۱ محاسبه کنید. همچنین متریک های هنری ـ کفورا و ریخت‌شناسی را برای این مدل طراحی محاسبه نمایید.

۱۹-۸ یک سیستم اطلاعاتی عظیم دارای ۱۱۴۰ پیمانه است. ۹۶ پیمانه، کارکردهای کنترلی و هماهنگی را بر عهده دارند و عملکرد ۴۹۰ پیمانه وابسته به پردازش قبلی است. سیستم حدود ۲۲۰ شیء داده‌ای را پردازش می‌کند که هر یک به‌طور متوسط سه صفت خاصه دارند. در این سیستم تعداد اقلام یکتای پایگاه داده ها ۱۴۰ قلم می باشد که مشتمل بر ۹۰ سگمنت مختلف خواهد بود. در نهایت، ۶۰۰ پیمانه از یک نقطه ورود و خروج برخوردارند. DSQI را برای این سیستم محاسبه کنید.

۱۹-۹ مقاله بیمان و لوت [BIE94] را مطالعه کرده و مثال کاملی بیاورید که محاسبه متریک چسبندگی و انسجام آنان را نشان دهد. اطمینان حاصل کنید که برش‌های داده‌ای، توکن ها و نشانه‌های داده‌ای، توکن های چسب و اثرچسب تعیین شده اند.

۱۹-۱۰. پنج پیمانه را در یک برنامه کامپیوتری موجود، برگزینید. با استفاده از متریک داما که در بخش ۱۹-۴-۲ شرح داده شد، مقدار پیوستگی را برای هر یک از پیمانه‌ها محاسبه نمایید.

۱۹-۱۱ یک ابزار نرم‌افزاری توسعه دهید که پیچیدگی سیکلوماتیک را برای یک پیمانه زبان برنامه‌نویسی محاسبه کند. انتخاب زبان با خودتان است.

۱۹-۱۲ یک ابزار نرم افزاری بسازید که پیچیدگی سیکلوماتیک را برای یک پیمانه زبان برنامه سازی، محاسبه نماید. زبان را خود انتخاب کنید.

۱۹-۱۳ یک ابزار نرم‌افزاری بسازید که میزان مناسب بودن طرح‌بندی را برای یک رابط گرافیک کاربر محاسبه کند. این ابزار باید شما را قادر سازد که هزینهٔ انتقال میان موجودیتهای طرح‌بندی را تخصیص دهید. (توجه: که اندازه جمعیت بالقوهٔ جایگزین های طرح‌بندی با رشد و افزایش تعداد موقعیت های توری (سیم بندی) به‌شدت رشد می‌یابد.)

۱۹-۱۴ ابزار نرم‌افزاری کوچکی بسازید که تحلیل هالستید را برای کد برنامه منبعی از زبان برنامه سازی منتخب خودتان به انجام رساند.

۱۹-۱۵ در متون تحقیق کنید و مقاله‌ای درباره ارتباط متریک هالستید و متریک مک کیب برای کیفیت نرم‌افزار بنویسید(آنگونه که با شمارش خطاها اندازه گیری می شوند). آیا داده‌ها قابل اعتنا هستند؟ رهنمودهایی برای استفاده از این متریک ها پیشنهاد کنید.

۱۹-۱۶ تحقیقی در خصوص مقالات جدید متریکهای خاص جهت یاری به طراحی مورد آزمون انجام دهید. یافته‌های خود را در کلاس ارائه کنید.

۱۹-۱۷ یک سیستم قدیمی ۹۴۰ پیمانه دارد. آخرین نسخه نیازمند تغییر ۹۰ پیمانه از مجموعه پیمانه هاست. به‌علاوه، ۴۰ پیمانه جدید اضافه شده و ۱۲ پیمانه نیز حذف شده‌اند. شاخص بلوغ نرم‌افزار را برای این سیستم محاسبه کنید.

فهرست منابع و مراجع

[BAS84] Basili, V.R. and D.M. Weiss, *itA* Methodology for Collecting Valid Software Engineering Data," *IEEE Trans. Software Engineering*, vol. SE-10, 1984, pp. 728-738.

[BIE94] Bieman, J.M. and L.M. ott, "Measuring Functional Cohesion," *IEEE Trans. Software Engineering*, vol. SE-20, no. 8, August 1994, pp. 308-320.

[BRI96] Briand, L.C., S. Morasca, and V.R. Basili, "Property-Based Software Engineering Measurement," *IEEE Trans. Software Engineering*, vol. SE-22, no. 1, *January* 1996, pp. 68-85.

[CAR90] Card, D.N. and R.L. Glass, *Measuring software Design Quality*, Prentice-Hall, 1990.

[CAV78] Cavano, *J.P.* and *J.A.* McCall, itA Framework for the Measurement of Software Quality, It *Proc. ACM Software Quality Assurance Workshop*, November 1978, pp. 133-139.

[CHA89] Charette, R.N., *Software Engineering Risk Analysis and Management* McGraw-Hill/Intertext, 1989.

[CURBO] Curtis, W. "Management and Experimentation in Software Engineering," *Proc. IEEE*, vol. 68, no. 9, September 1980.

[DAV93] Davis, A., et al., "Identifying and Measuring Quality in a Software Requirements Specification, *proc. First Ind. software Metrics Symposium*, IEEE, Baltimore, MD, May 1993, pp. 1.4 1-152.

[DEM81] DeMillo; R.A. and R.J. Lipton, "Software Project Forecasting," in *Software Metrics (A.J.* Perlis, F.G. Sayward, and M. Shaw, eds.), MIT Press, 1981, pp. 77-89.

[DEM82] DeMarco, T., *Controlling Software Projects*, Yourdon Press, 1982.

[DHA95] Dhama, H., "Quantitative Models of Cohesion and Coupling in Software," *journal of Systems and Software*, vol. 29, no. 4, April 1995.

Ejiogu, L., *Software Engineering with Formal Metrics*, QED Publishing, 1991. [EJI91]

[FEL89] Felican, L. and G. Zalateu, "Validating Halstead's Theory for Pascal Programs," *IEEE Trans. Software Engineering*, vol. SE-15, no. 2, December 1989, pp. 1630-1632.

[FEN91] Fenton, N., *Software Metrics*, Chapman and Hall, 1991.

[FEN94] Fenton, N., "Software Measurement: A Necessary Scientific Basis," *IEEE Trans. Software Engineering*, vol. SE-20, no. 3, March 1994, pp. 199-206.

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

[HAL77] Halstead, M., *Elements of Software Science*, North-Holland, 1 977.

[HENBl] Henry, S. and D. Kafura, "Software Structure Metrics Based on Information Flow," *IEEE Trans. Software Engineering*, vol. SE- 7, no. 5, September 1981, pp. 510-518.

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

{IEE94] *Software Engineering Standards*, 1994 edition, IEEE, 1994.

[KYB84] Kyburg, H.E., *Theory and Measurement*, Cambridge university Press, 1984.

[MCC76] McCabe, *T.J.*, "A Software Complexity Measure, *IEEE Trans. Software Engineering*, vol. SE-2, December 1976, pp. 308-320.

[MCC77] McCall, J., P. Richards, and G. Walters, "Factors in Software Quality," three

volumes, NTIS AD-A049-014, 015, 055, November 1977.
[MCC89] McCabe, T.J. and C.W. Butler, "Design Complexity Measurement and Testing," *CACM*, vol. 32, no. 12, December 1989, pp. 1415-1425.
[MCC94] McCabe, T.J. and A.H. Watson, "Software Complexity," *Crosstalk*, vol. 7, no. 12, December 1994, pp. 5-9.
[NIE94] Nielsen, J., and J. Levy, "Measuring Usability: Preference vs. Performance," *CACM*, vol. 37, no. 4, April 1994, pp. 65-75.
[ROC94] Roche, *J.M.*, "Software Metrics and Measurement Principles," *Software Engineering Notes*, ACM, vol. 19, 'no. I, January 1994, pp. 76-85.
[SEA93] Sears, A., "Layout Appropriateness: A Metric for Evaluating User Interface Widget Layout, *IEEE Trans. Software Engineering*, vol. SE-19, no. 7, July 1993, pp. 707-719.
[USA87] *Management Quality Insight*, AFCSP 800- I 4 (U.S. Air Force), January 20, 1987.
[ZUS90] Zuse, H., *Software Complexity: Measures and Methods*, DeGruyter, 1990.
[ZUS97] Zuse, H., *A Framework of Software Measurement*, DeGruyter, 1997.

خواندنیهای دیگر و منابع اطلاعاتی

There are a surprisingly large number of books that are dedicated to software metrics, although the majority focus on process and project me tries to the exclusion of technical metrics. Zuse [ZUS97] has written the most thorough treatment of technical metrics published to date.

Books by Card and Glass [CAR90], Zuse [ZUS90], Fenton {FEN91], Ejiogu [EJI91], Moeller and paulish *(Software Metrics*, Chapman and Hall, 1993), and Hetzel [HET93] all address technical metrics in some detail. Oman and Pfleeger *(Applying Software Metrics*, IEEE Computer Society Press, 1997) have edited an anthology of important papers on software metrics. In addition, the following books are worth examining:

Conte, S.D., H.E. Dunsmore, and v.y. Shen. *Software Engineering Metrics and Models*, Benjamin/Cummings, 1984.

Fenton, N.E. and S.L. Pfleeger, *Software Metrics: A Rigorous and Practical Approach*, 2nd ed., PWS Publishing Co., 1998.

Grady, R.B. *Practical Software Metrics for Project Management and Process Improvement*, Prentice-Hall, 1992.

Perlis, A., et al., *Software Metrics: An Analysis and Evaluation*, MIT Press, 1981.

Sheppard, M., *Software Engineering Metrics*, McGraw-Hill, 1992.

The theory of software measurement is presented by Denvir, Herman, and Whitty in an edited collection of papers *(Proceedings of the International BCS-FACS Workshop: Formal Aspects of Measurement*, Springer-Verlag, 1992). Shepperd *(Foundations of software Measurement*, Prentice-Hall, 1996) also addresses measurement theory in some detail.

A comprehensive summary of dozens of useful software metrics is presented in [IEE94]. In general, a discussion of each metric has been distilled to the essential "primitives" (measures) required to compute the metric and the appropriate relationships to effect the computation. An appendix provides discussion and many references.

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

http://www.mhhe.com/engcs/compsci/pressman/resources/tech-metrics.mhtml