فصل ۸ تضمین کیفیت نرم افزار

مفاهیم کلیدی (مرتب بر حروف الفبا)

ایزو ۹۰۰۰ ، ایمنی نرم افزار ، بازبینی فنی رسمی ، پوکا- یوک ، تشدید نقص ، تضمین کیفیت نرم افزار ، تضمین کیفیت نرم افزار آماری ، طرح تضمین کیفیت نرم افزار ، فعالیتهای تضمین کیفیت نرم افزار ، قابلیت تغییرپذیری کیفیت ، هزینه های کیفیت

KEY CONCEPTS

Defect amplification , formal technical reviews , IS09000 , poka yoke , quality , quality costs , software safety , SQA , SQA activities , SQA plan , statistical SQA , Variation

نگاه اجمالی

تضمین کیفیت نرم افزارچیست؟ تنها گفتن این‌که کیفیت نرم‌افزاری مهم است کافی نمی‌باشد، شما باید (۱) باید به روشنی بیان کنید که وقتی می‌گویید "کیفیت نرم‌افزار" منظورتان چیست، (۲) یک مجموعه فعالیت‌هایی را به‌وجود آورید که به تضمین کیفیت بالای هر محصول مهندسی نرم‌افزاری کمک می‌کنند، (۳) فعالیت‌های تضمین کیفیت را بر روی هر پروژهٔ نرم‌افزاری‌ای اجرا کنید، (۴) متریک‌هایی را برای توسعهٔ راهبردهایی برای بهبود فرآیند نرم‌افزاری خود به‌کار ببرید، و در نتیجه، کیفیت محصول پایانی را بالا ببرید.

چه کسی آن را انجام می‌دهد؟ همه کسانی که در روند مهندسی نرم‌افزار درگیرند، مسئول کیفیت آن هستند.

دلیل اهمیت آن چیست؟ شما می‌توانید آن را درست انجام دهید، یا می‌توانید آنها چندین بار دوباره انجام دهید. اگر یک تیم نرم‌افزاری در تمام فعالیت‌های مهندسی نرم‌افزار بر کیفیت تأکید داشته باشد، دوباره کاری‌هایی که باید انجام گیرند کاهش می‌یابد. و همین منجر به هزینه‌های پایین‌تر و از آن مهم‌تر زمان ارائه به‌بازار را بهبود می‌دهد.

چه قدم‌هایی باید برداشته شود؟ قبل از این‌که فعالیت‌های تضمین کیفیت نرم‌افزار شروع شوند، تعریف "کیفیت نرم‌افزار" در سطوح انتزاعی مختلف لازم است. زمانی‌که شما فهمیدید که کیفیت چیست، یک تیم نرم‌افزاری باید یک دسته از فعالیت‌های SQA را شناسایی کند، که نقایص محصولات کاری را به‌موقع رفع نماید.

حاصل کار چیست؟ یک طرح تضمین کیفیت نرم‌افزار برای تعریف راهبرد **SQA** تیم نرم‌افزاری به‌وجود می‌آید. در خلال تجزیه و تحلیل، طراحی و برنامه‌نویسی، محصول کار **SQA** اولیه، گزارش خلاصه شدهٔ بررسی فنی و رسمی است. در خلال آزمون، طرح‌های آزمون و رویه‌ها تولید می‌شوند. محصولات کاری دیگری که با بهبود فرآیند همراهند نیز، ممکن است به‌وجود آیند.

چگونه مطمئن شوم که کار را درست انجام داده‌ام؟ خطاها را قبل از این‌که تبدیل به نقص شوند پیدا کنید؛ یعنی تلاش کنید کارآیی رفع نقص خود را (فصل ۴ و ۷) افزایش دهید، و از این راه دوباره‌کاری‌هایی را که تیم نرم‌افزاری شما باید انجام دهند کاهش دهید.

رهیافت مهندسی نرم‌افزاری که در این کتاب توصیف شد در جهت یک هدف پیش می‌رود: تولید نرم‌افزار با کیفیت بالا. اما بسیاری از خوانندگان با این سئوال دست بگریبانند: "کیفیت نرم‌افزاری چیست؟"

فیلیپ کروسبای **[CRO79]**[1] در کتاب برجسته خود دربارهٔ کیفیت، جواب خاصی به این سئوال می‌دهد:

مشکل مدیریت کیفیت چیزی که مردم دربارهٔ آن نمی‌دانند نیست. مسئله چیزی است که آنها فکر می‌کنند می‌دانند ...

از این جهت، کیفیت مشترکات بسیاری با جنسیت دارد. هر کسی به‌دنبال آن است (البته تحت شرایط خاصی). همه فکر می‌کنند آن را می‌فهمند (اگر چه نمی‌خواهند آن را توضیح دهند). همه فکر می‌کنند که اجرا کردن تنها پیروی کردن از یک گرایش طبیعی است (به‌هر حال، ما باید با کسی همراه شویم). و البته بیشتر مردم احساس می‌کنند که مشکلات موجود در این زمینه‌ها توسط مردم دیگر به‌وجود آمده‌اند (گو این‌که تنها آنها فرصت انجام درست کارها را دارند).

بعضی برنامه‌نویسان نرم‌افزاری هم‌چنان به این عقیده پایبندند که کیفیت نرم‌افزاری چیزی است که شما پس از به‌وجود آوردن برنامه نگران آن می‌شوید. حقیقتی جز این وجود ندارد! تضمین کیفیت نرم‌افزاری[2] (**SQA**) یک فعالیت چتری پوششی است (فصل ۲) که در سراسر فرآیند نرم‌افزاری را دربرمی‌گیرد.

(SQA) شامل این موارد می‌شود: (۱) یک رهیافت مدیریت کیفیت، (۲) فناوری مهندسی مؤثر (روش‌ها و ابزارها)، (۳) بررسی‌های فنی منظم که در سراسر روند فرآیند نرم‌افزاری اجرا می‌گردد؛ (۴) یک راهبرد آزمون چند لایه‌ای، (۵) کنترل مستندسازی نرم‌افزار و تغییراتی که در آن به‌وجود می‌آید، (۶) روالی برای اطمینان مطابقت با استانداردهای توسعهٔ نرم‌افزاری (هرگاه قابل اجرا باشند)، و (۷) و مکانیزم‌های سنجش و گزارش.

1.Crosby,P

2.Software Quality Assurance

در این فصل، ما به موضوعات مربوط به سنجش و نیز فعالیت‌های روند ـ خاصی که یک سازمان نرم‌افزاری را قادر می‌سازد اطمینان حاصل کند که "کار صحیح در زمان مقتضی و به شیوهٔ درست" انجام می‌گیرد، می‌پردازیم.

۸-۱ مفاهیم کیفیت[1]

گفته شده است که هیچ دو دانهٔ برفی یکسان نیستند. مطمئناً وقتی که ما به تماشای بارش برف می‌پردازیم، تصور این‌که دانه‌های برف همگی با هم متفاوتند مشکل است، پس اجازه بدهید هر دانهٔ برفی دارای ساختار منحصربه‌فرد خودش باشد. به‌منظور مشاهدهٔ تفاوت بین دانه‌های برف، شاید به‌وسیلهٔ یک ذره‌بین، ما باید نمونه‌ها را از نزدیک مورد آزمون قرار دهیم. در واقع، هر چه از نزدیک‌تر آنها را مشاهده کنیم، تفاوت‌های بیشتری را خواهیم یافت.

نقل قول

مردم سرعت انجام عمل شما را فراموش خواهند نمود، اما کیفیت کار شما را همواره به یاد خواهند داشت. هاوارد نیوتن

این پدیده، گوناگونی بین نمونه‌ها[2]، در مورد تمام تولیدات دست بشر و نیز مخلوقات طبیعی صادق است. به‌عنوان مثال، اگر دو برد مدار از فاصله نزدیک مورد آزمون قرار گیرند، ما ممکن است مشاهده کنیم که باریکه‌های مسی روی برد از لحاظ شکل هندسی، مکان و ضخامت، اندکی با هم متفاوتند. همچنین، موقعیت و قطر سوراخ‌هایی نیز که بر روی برد ایجاد شده‌اند با هم فرق می‌کنند.

کنترل تنوع[3] قسمت اصلی کنترل کیفیت است. یک تولیدکننده حتی وقتی به کاری ساده چون کپی کردن دیسکت می‌پردازد، تلاش دارد تا گوناگونی در بین تولیدات به حداقل برساند. مطمئناً این می‌تواند یک مشکل باشد ـ کپی کردن دیسکت یک عملیات تولیدی پیش پا افتاده است و ما می‌توانیم اطمینان دهیم که همواره کپی‌های دقیقی از نرم‌افزار تولید می‌شود.

از طرف دیگر آیا می‌توانیم این کار را انجام دهیم؟ باید مطمئن شویم که شیارها با تولرانس ویژه‌ای بر روی دیسکت‌ها قرار گرفته‌اند، به‌طوری‌که اکثریت کلی دیسک‌گردان‌ها بتوانند دیسک‌ها را بخوانند. همچنین باید مطمئن شویم که فلو مغناطیسی لازم برای تمایز صفر از یک، هدهای خواندن/ نوشتن کافی است. دستگاه‌های تکثیر دیسک قادر به انجام این کارند، و این کار را انجام می‌دهد، اما از لحاظ تولرانس آنها نیز گرفتار تغییر و نوسان می‌شوند. بنابراین حتی یک فرآیند "ساده" مانند تکثیر دیسک ممکن است به‌دلیل تنوع بین نمونه‌ها با مشکل مواجه شود.

کنترل تغییرات کلید کیفیت بالای محصول می باشد. در خصوص نرم افزار، ما تلاش می کنیم که تغییرات فرآیندی که مشغول آن هستیم را کنترل نماییم، همین طور منابع مورد استفاده، و خصوصیات کیفی محصول نهایی را.

اما چگونه این پدیده در مورد کار نرم‌افزاری اعمال می‌گردد؟ چگونه ممکن است یک سازمان توسعهٔ نرم‌افزاری نیاز به کنترل تفاوت‌ها پیدا کند؟ در هر پروژه‌ای نسبت به پروژهٔ دیگر، ما تمایل داریم تفاوت بین

---

۱. این بخش توسط مایکل استوسکی تحریر گردیده است، که مطابق اصول ایزو ۹۰۰۰ و استاندارد ایزو ۹۰۰۱ شکل یافته است. کتابهای کاربردی برای مهندسی نرم افزار و یک فیلم ویدئویی که توسط راجر اس. پرسمن و تشکیلاتش، (با کسب اجازه)تهیه شده است.

2.variation between samples

3.Variation control

منابع پیش‌بینی شده برای کامل کردن یک پروژه و منابع واقعی‌ای که به‌کار برده شده‌اند، شامل به‌کارگماری کارکنان، تجهیزات و زمان تقدیمی را به حداقل برسانیم. معمولاً، ما تمایل داریم مطمئن شویم که برنامهٔ آزمون‌ها درصد شناخته شده‌ای از نرم‌افزار را در هر نشر، نسبت به دیگری در برمی‌گیرد. ما نه‌تنها می‌خواهیم تعداد نقایصی را که وارد محیط ما شده‌اند به حداقل برسانیم، بلکه تمایل داریم مطمئن شویم که واریانس تعداد اشکالات نیز درهر نشر، نسبت به دیگری به حداقل رسیده است. (احتمالاً مشتری‌های ما اگر نقایص سومین نشر یک تولید، ده برابر نقایص نشر قبلی باشد ناراحت خواهند شد). ما دوست داریم تفاوت‌های موجود در سرعت و دقت پاسخ‌های حمایتی فوری خود به مشکلات مشتری را به حداقل کاهش دهیم.

۸-۱-۱ کیفیت

واژه‌نامهٔ هریتیج آمریکا[1] کیفیت[2] را به‌عنوان "ویژگی یا خصیصهٔ چیزی" تعریف می‌کند. کیفیت به‌عنوان خصیصهٔ چیزی به ویژگی‌های قابل سنجش چیزی اشاره دارد ـ چیزهایی که ما می‌توانیم آنها را با استانداردهای شناخته شده‌ای چون طول، رنگ، خواص الکتریکی، چکش‌خواری و غیره مقایسه کنیم. با این همه مشخص کردن نرم‌افزار، که عمدتاً جوهری فکری است، بیشتر از اجسام فیزیکی مشکل می‌باشد.

علی‌رغم این، مقیاس‌هایی برای ویژگی‌های یک برنامه وجود دارد. این ویژگی‌ها عبارتند از: پیچیدگی سیلکومتیک، یکپارچگی، تعداد امتیازات کارکردی، خطوط برنامه‌نویسی، و بسیاری دیگر که در فصل‌های ۱۹ و ۲۴ بحث شده‌اند. وقتی ما چیزی را بر مبنای ویژگی‌های قابل سنجش آن می آزماییم، ممکن است با دو نوع کیفیت روبه‌رو شویم: کیفیت طراحی و کیفیت تطابق.

کیفیت طراحی[3] مربوط به آن ویژگی‌هایی است که طراح به چیزی اختصاص داده است. درجهٔ مواد، تولرانس‌ها و خصوصیات اجرایی همه مربوط به کیفیت طراحی هستند. با به‌کار بردن موادی با درجهٔ عالی‌تر، تولرانس‌های فشرده‌تر و سطوح اجرایی معین‌تر، کیفیت طراحی یک محصول افزایش می‌یابد، البته اگر آن محصول مطابق آن ویژگی‌ها تولید گردد.

کیفیت تطابق[1] میزان رعایت مشخصات طراحی به هنگام ساخت محصول است. در این‌جا نیز، با بالا رفتن میزان تطابق، کیفیت تطابق محصول بالاتر می‌رود.

در توسعهٔ نرم‌افزاری، کیفیت طراحی شامل نیازمندی‌ها، خصوصیات، و طرح سیستم می‌شود. کیفیت تطابق موضوعی است که در وحلهٔ نخست بر پیاده‌سازی، تأکید دارد. اگر در پیاده‌سازی طراحی رعایت شود و سیستم حاصل به نیازمندیها و اهداف عملکردی‌اش نائل شود، کیفیت تطابق بالاست.

نقل قول

زمان کمتری نیاز است تا کاری را به خوبی انجام دهید نسبت به زمانی که باید سپری کنید تا اشتباهات آن را توجیه نمایید. هنری وارس ورث لانگ فرد

1.American Heritage Dictionary

2.quality

3.Quality of design

اما آیا کیفیت طراحی و کیفیت تطابق تنها موضوعاتی هستند که مهندسان نرم‌افزار باید مورد ملاحظه قرار دهند؟ روبرت گلاس [GLA98][2] بیان می‌دارد که رابطه "ادراکی" دیگری در کار است:

+ کیفیت خود + تولید مورد قبول = رضایت مصرف‌کننده

تحویل بر طبق بودجه و برنامه

در خط پایانی، گلاس اظهار می‌دارد که کیفیت مهم است، اما اگر مصرف‌کننده راضی نباشد، در واقع امور دیگر، بی‌فایده‌اند. دی‌مارکو [DEM 99][3] با بیان این مطلب عقیدهٔ گلاس را تقویت می‌کند: "کیفیت یک محصول تابعی از تغییری است که آن محصول در جهت بهتر کردن جهان به‌وجود آورده است." این دید نسبت به کیفیت معتقد است که اگر یک محصول نرم‌افزاری منفعت عمده‌ای برای مصرف‌کنندگان نهایی‌اش فراهم آورد، آنها ممکن است نقایص اتفاقی یا مسایل عملکردی را تحمل کنند.

پرسش

کنترل کیفیت نرم افزار چیست؟

۸-۱-۲ کنترل کیفیت[4]

تغییر کنترل ممکن است با کنترل کیفیت برابر گردد. اما چگونه ما به کنترل کیفی دست می‌یابیم؟ کنترل کیفیت عبارت است از مجموعهٔ بازرسی‌ها، بررسی‌ها و آزمونهایی که در روند فرآیند نرم‌افزاری انجام می‌گیرد و هدف آن تضمین تطبیق تولید هر کالا با نیازهایی است که به‌خاطر آن تولید می‌شود. کیفیت کنترل شامل یک حلقهٔ بازخوردی است که محصول کاری را به‌وجود آورده است. ترکیب سنجش و بازخورد به ما اجازه می‌دهد تا زمانی‌که تولیدات کاری فاقد مشخصات خود هستند به تنظیم امور بپردازیم. این رهیافت کنترل کیفیت را بخشی از روند تولید می‌داند.

فعالیت‌های کنترل کیفیت ممکن است کاملاً خودکار، کاملاً دستی، یا ترکیبی از ابزارهای خودکار و تعامل انسانی باشد. یک مفهوم کلیدی در کنترل کیفیت این است که تمام محصولات کاری دارای خصوصیات تعریف شده و قابل اندازه‌گیری هستند، که با آنها ممکن است ما خروجیهای هر فرآیندی را مقایسه کنیم. به‌منظور به حداقل رساندن نقایص تولیدی وجود حلقهٔ بازخوردی الزامی است.

ارجاع به وب

منابع متنوع و گسترده ای در خصوص کیفیت نرم افزار ها با مراجعه به آدرس زیر بیابید :
www.qualitytree.com/links/links/html

۸-۱-۳ تضمین کیفیت

تضمین کیفیت[5] شامل حسابرسی (وارسی) و گزارش کارکردهای مدیریت است. هدف تضمین کیفیت، فراهم کردن اطلاعات لازم برای مدیریت است که به‌وسیلهٔ آنها از کیفیت تولید آگاه می‌شود و در نتیجه اطمینان کامل حاصل می‌کند که کیفیت تولید در جهت برآوردن اهداف است. البته اگر

1.Quality of conformance
2.Glass,R.
3.DeMarco,T.
4.Quality control
5.Quality assurance

اطلاعاتی که از راه تضمین کیفیت حاصل گشته‌اند مشکلاتی را شناسایی کنند، این وظیفهٔ مدیریت است که مشکلات را مورد توجه قرار دهد و منابع لازم برای حل مشکلات مربوط به کیفیت را به‌کار اندازد.

۸-۱-۴ هزینه کیفیت

پرسش

اجزاء و مولفه های هزینه کیفیت کدامها می باشند؟

هزینهٔ کیفیت[1] شامل تمام هزینه‌هایی است که در جهت نیل به کیفیت یا اجرای فعالیت‌های مربوط به کیفیت به بار می‌آیند. مطالعاتی در مورد هزینهٔ کیفیت انجام شده است که هدف آنها فراهم کردن بستری برای هزینهٔ کیفیت فعلی، شناسایی فرصت‌هایی برای کاهش هزینهٔ کیفیت و فراهم کردن یک اساس طبیعی برای مقایسه می‌باشد. اساس طبیعی کردن این کار تقریباً همیشه دلار است. اگر یکبار هزینه‌های کیفیت را بر پایهٔ دلار طبیعی کنیم، داده‌های لازم برای ارزشیابی فرصت‌هایی که برای بهبود پیشرفت‌هایمان وجود دارد، در اختیار خواهیم داشت. همچنین، ما می توانیم اثر تغییرات را بر مواردی که بر پایهٔ دلار استوارند ارزشیابی کنیم. هزینه‌های کیفیت[2] ممکن است به هزینه‌هایی که مربوط به پیشگیری، ارزیابی و شکست می‌باشند، تقسیم کرد. هزینه‌های پیشگیری[3] عبارتند از:

- برنامه‌ریزی کیفیت
- بررسی‌های منظم فنی
- تجهیزات آزمون
- آموزش

هزینه‌های ارزیابی[4] شامل فعالیت‌هایی برای آگاهی از وضعیت تولید در "آغاز" هر فرآیند می‌باشد. مثال‌هایی از هزینه‌های ارزیابی عبارتند از:

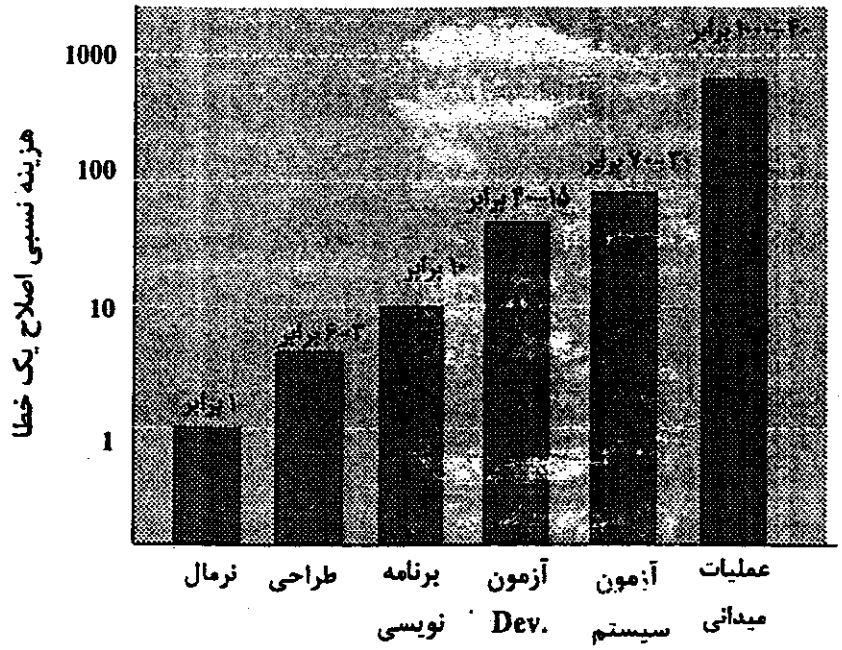
- بازرسی درون ـ مرحله‌ای و بین ـ مرحله‌ای
- تعمیر و نگهداری تجهیزات
- انجام آزمون

1.Cost of quality

2.Quality costs

3. Prevention costs

4.Appraisal costs



شکل ۸-۱ هزینه نسبی اصلاح یک خطا

هزینه‌های شکست[1] آنهایی هستند که در صورت معلوم نشدن نقصی قبل از تحویل محصول به مشتریان، مرتفع خواهند شد. هزینه‌های شکست ممکن است به هزینه‌های شکست درونی و هزینه‌های شکست بیرونی تقسیم گردند. هزینه‌های شکست داخلی[2] عبارتند از:

- کار دوباره
- تعمیر
- تجزیه و تحلیل وضعیت شکست (و یافتن دلائل نا کارآمدی- م.)

هزینه‌های شکست بیرونی[3] آنهایی هستند که مربوط به نقایصی هستند که بعد از ارسال محصول به مشتری پیدا شده‌اند. نمونه‌هایی از هزینه‌های شکست داخلی عبارتند از:

- هزینهٔ رفع و رجوع شکایات و دعاوی حقوقی
- بازگشت و تعویض کالا
- پشتیبانی و کمک‌های حمایتی
- انجام تعهدات تضمین داده شده (مطابق قراردادها – م.)

همان‌طور که انتظار می‌رفت، هزینهٔ نسبی پیدا کردن و رفع یک نقص با حرکت ما از پیش‌گیری

توصیه

از افزایش هزینه های پیشگیری موثر هراسی نداشته باشید. آسوده باشید و مطمئن که این نوعی سرمایه گذاری می باشد که بازگشتی عالی خواهد داشت.

1.Failure costs

2.Internal failure costs

3.External failure costs

به تشخیص و سپس به هزینه‌های شکست داخلی و شکست خارجی به میزان قابل ملاحظه‌ای افزایش می‌یابد. تصویر ۸-۱، بر اساس داده‌هایی که بوهم [BOE81][1] و دیگران جمع‌آوری کرده‌اند این پدیده را نشان می‌دهد.

تست و آزمون ضروری است. اما راهی گران قیمت برای یافتن خطاست. اگر در مراحل اولیه فرآیند وقت بیشتری را اختصاص دهید، هزینه آزمون و رفع خطا را به طور قابل ملاحظه ای کاهش خواهید داد.

داده‌های جالبی که توسط کاپلان و همکارانش کلارک و تانگ [KAP95][2] گزارش شده‌اند آمارهای هزینهٔ قبلی را تقویت می‌کنند و بر مبنای کار انجام شده در تسهیلات توسعهٔ روجستر آی‌بی‌ام قرار دارند:

بعد از صرف ۷۰۵۳ ساعت برای بررسی ۲۰۰/۰۰۰ خط برنامه‌نویسی معلوم گردید که از ۳۱۱۲ نقص بالقوه جلوگیری شده است. اگر حقوق یک برنامه‌نویس را ۴۰/۰۰ دلار در ساعت در نظر گیریم، هزینهٔ کلی پیش‌گیری از ۳۱۱۲ نقص برابر خواهد بود با ۲۸۲/۱۲۰ دلار یا تقریباً ۹۱/۰۰ دلار برای هر نقص.

این اعداد را با هزینهٔ رفع نقص، بعد از این‌که کالا برای مشتری ارسال می‌شود مقایسه کنید. فرض کنید که هیچ‌گونه بازرسی‌ای در کار نبوده است، اما برنامه‌نویس دقت بیشتری مبذول داشته است و از هر ۱۰۰۰ خط برنامه‌نویسی [به میزان زیادی بیشتر از میانگین صنعتی] تنها یک مورد نقص‌دار وارد محصول حمل شده گردیده است. این بدان معنی خواهد بود که هنوز ۲۰۰ مورد نقص‌دار در محیط باقی خواهد ماند. با یک هزینهٔ تخمینی ۲۵/۰۰۰ دلاری برای هر زمینهٔ نقص‌دار، هزینهٔ کلی برابر خواهد بود با ۵ میلیون دلار، یا تقریباً ۱۸ برابر بیشتر از هزینهٔ کلی کار پیشگیری.

درست است که IBM نرم‌افزارهایی تولید می‌کند که به‌وسیلهٔ صدها هزار نفر به‌کار برده می‌شوند و هزینه رفع نقص آن بیشتر از هزینه‌های سازمان‌های نرم‌افزاری است که سیستم‌های سفارشی تولید می‌کنند. اما این به هیچ‌وجه نتایجی را در بالا بدان‌ها اشاره شد بی‌اثر نمی‌سازد. حتی اگر سازمان نرم‌افزاری متوسطی دارای هزینه‌هایی برابر با ۲۵درصد هزینه‌های IBM باشد (بیشتر وقت‌ها نظری درباره این‌که هزینه‌هایشان چه هستند وجود ندارد)، هزینه‌های صرفه‌جویی شدهٔ مربوط به کنترل کیفیت و فعالیت‌های تضمینی قابل ملاحظه‌اند.

مدیریت کیفیت جامع (TQM) می تواند برای نرم افزار کامپیوتر به کار رود. رهیافت TQM بر پیشرفت پیوسته فرآیند تاکید می کند.

۸-۲ حرکت کیفی

امروزه مدیران مسئول کمپانی‌های سراسر جهان صنعتی فهمیده‌اند که تولید با کیفیت بالا منجر به صرفه‌جویی در هزینه‌ها و محصول پایانی بهتر می‌شود. با این‌حال، همواره چنین چیزی اتفاق نمی‌افتد. حرکت کیفی در دههٔ چهل قرن بیستم با کار نخستین دبلیو ادوارد دمینگ [DEM86][3] آغاز گردید و اولین آزمون واقعی آن در ژاپن صورت گرفت. ژاپنی‌ها با مبنا قرار دادن عقاید دمینگ، رهیافت نظام‌مندی به‌وجود آوردند که هدف آن از بین بردن علت نقص‌های موجود در محصولات بود، در تمام سال‌های دههٔ

1.Boehm,B.
2.Kaplan,C.
3.Deming,W.E.

۷۰ و ۸۰ قرن بیستم کار آنها به جهان غرب منتقل شد و به آن نام‌هایی چون "مدیریت کیفی کلی (TQM)" داده شد.[1] اگر چه کمپانی‌ها و نویسندگان مختلف اصطلاحات متفاوتی در این زمینه به‌کار می‌برند، اما در هر برنامهٔ مدیریت کیفی کلی مطلوبی، معمولاً با یک توالی چهار مرحله‌ای اصلی به‌عنوان زیر بنا روبه‌رو می‌شویم.

مرحلهٔ اول، کایزن[2] گفته می‌شود و اشاره به سیستمی با روند پیشرفت دایمی دارد. هدف مرحلهٔ کایزن به‌وجود آوردن فرآیندی (در این مورد خاص، روند نرم‌افزاری) قابل مشاهده، تکرارپذیر و سنجش‌پذیر است.

مرحلهٔ دوم، که تنها پس از به نتیجه رسیدن مرحلهٔ کایزن آغاز می‌گردد، آتاریما هینشیتسو[3] گفته می‌شود. این مرحله به بررسی عوامل پنهانی که بر روند کار تأثیر می‌گذارد می‌پردازد و تلاش می‌کند تأثیر آنها بر روند کار را خوش‌بینانه نماید. به‌عنوان مثال، فرآیند نرم‌افزاری ممکن است تحت‌تأثیر چرخش‌های بزرگی که در کارکنان شرکت انجام می‌گیرد قرار گیرد، که خود آنها تحت‌تأثیر سازمان‌بندی‌های دوبارهٔ داخل یک شرکت به‌وجود آمده‌اند. آتاریما هینشیتسو مدیریت را در جهت بروز تغییراتی متناسب با سازمان‌بندی جدید هدایت خواهد کرد.

در حالی‌که دو مرحلهٔ اول بر فرآیند تأکید دارند، مرحلهٔ بعد که کانسئی[4] گفته می‌شود (به‌معنای "حواس پنج‌گانه") بر مصرف‌کنندهٔ محصول تمرکز می‌کند (در این‌مورد خاص، نرم‌افزار) در اساس، با بررسی چگونگی کاربرد محصول از سوی مصرف‌کننده کانسئی به بهبود بخشیدن خود محصول می‌پردازد، و بطور بالقوه نیز فرآیندی را که به‌وجود آورده است، تقویت می‌کند.

و بالاخره، مرحله‌ای که میریوکوتکی هینشیتسو[5] گفته می‌شود، توجه مدیریت را به فردای تولید کنونی معطوف می‌کند. این مرحله مرحله‌ای تجارت‌گراست که به‌دنبال یافتن فرصت در زمینه‌های مربوط است. و با مشاهدهٔ کاربرد کالا در بازار قابل شناسایی‌اند در جهت نرم‌افزار، میریوکوتکی هینشیتسو می‌تواند تلاشی برای آشکار کردن تولیدات و کاربردهای نو و سودآور در نظر گرفته شود که پیامد سیستم‌های مبتنی بر کامپیوتر موجود می‌باشند.

در بیشتر شرکت‌ها توجه به کایزن باید در اولویت باشد. تا زمانی که یک فرآیند نرم‌افزاری بالنده (فصل ۲) به‌وجود نیامده باشد، امکان اندکی برای حرکت به مرحلهٔ بعد وجود دارد.

ارجاع به وب

منابع متنوع و گسترده ای برای بهبود پیوسته فرآیند و مدیریت کیفیت جامع (TQM) را می توانید در آدرس زیر بیابید :
www.deming.eng.cl
emon.edu/

---

۱. برای آنکه از مدیریت کیفیت جامع، به درکی کامل برسید، مراجعه به [ART92] توصیه می شود و نیز [KAP95] برای توضیح استفاده از معیارهای بالدریج در جهان نرم افزار.

2.kaizen

3.atarimae

4.hinshitsu

5.Atarimae hinshitsu

۸-۳ تضمین کیفیت نرم افزار

حتی مهجورترین سازندگان نرم‌افزار بر این باورند که نرم‌افزار کیفیت بالا، هدفی اساسی است. اما ما چگونه کیفیت را تعریف می‌کنیم؟ زمانی شخصی گفت: "هر برنامه‌ای دارای چیز خوبی هست، فقط ممکن است آن چیزی نباشد که ما از آن می‌خواهیم."

در زمینهٔ کیفیت نرم‌افزاری تعریف‌های بسیاری ارائه شده است. تعریف کیفیت نرم‌افزاری با توجه به اهداف ما عبارت است:

تطابق با نیازهای عملکردی و کارکردی که به وضوح بیان شده‌اند، استانداردهای توسعه‌ای که بی‌هیچ ابهامی ثبت گردیده‌اند، و ویژگی‌های تردیدناپذیری که وجود آنها در هر نرم‌افزار حرفه‌ای توسعه یافته‌ای، انتظار می‌رود.

تردید اندکی وجود دارد که تعریف فوق قابل تعدیل یا بسط باشد. در واقع تعریف قطعی کیفیت نرم‌افزاری همواره می‌تواند موضوع جدل باشد. با توجه به اهداف این کتاب، از تعریف فوق برای تأکید بر سه نکته مهم استفاده می‌شود:

پرسش

چه هنگام و با چه هدفی بازبینی های رسمی را بکار می بریم ؟

۱- نیازمندیهای نرم‌افزاری بنیانی است که با توجه به آن کیفیت سنجیده می‌شود. فقدان هم‌سویی با نیازمندیها فقدان کیفیت است.

۲- استانداردهای خاصی یکسری معیارهای توسعه را تعریف می‌کنند که مسیر مهندسی نرم‌افزار را مشخص می‌کند. اگر این معیارها دنبال نگردند، تقریباً به‌طور حتم، فقدان کیفیت رخ خواهد نمود.

۳- یکسری نیازمندیهای ضمنی وجود دارد که اغلب بیان نشده باقی می‌مانند (مثلاً تمایل به استفادهٔ آسان و نگهداری مطلوب). اگر نرم‌افزار با نیازمندیهای آشکارش سازگار باشد، اما نیازمندیهای ضمنی خود را برآورده نسازد، کیفیت نرم‌افزار مورد شک قرار می‌گیرد.

نقل قول

ما خبط و خطاهای بسیار زیادی داریم. بوگی برا

۸-۳-۱ موضوعات زمینه

پیشینهٔ تضمین کیفیت در توسعهٔ نرم‌افزار، موازی با پیشینهٔ کیفیت در ساخت سخت‌افزار است. در خلال اولین روزهای به‌وجود آمدن کامپیوتر (دههٔ ۵۰ و ۶۰ قرن بیستم)، کیفیت تنها وظیفهٔ برنامه‌نویس بود. استانداردهای تضمین کیفیت برای نرم‌افزار در دههٔ ۷۰ برای نخستین‌بار در قراردادهای نظامی توسعهٔ نرم‌افزار به‌وجود آمد و سپس به سرعت به توسعهٔ نرم‌افزار در جهان تجارت گسترش یافت. [IEE 94][1] با بسط تعریف ارائه شده قبلی، کیفیت نرم‌افزار "الگوی نظام‌مند و برنامه‌ریزی شدهٔ فعالیت‌هاست."

ارجاع به وب

یک آموزش عمیق و طیف گسترده ای از منابع مدیریت کیفیت در آدرس زیر وجود دارد :
www.
Manarngement.gov

1.Software Engineering Standardards

[SCH98][1] که برای تضمین کیفیت نرم‌افزار الزامی‌اند. شاید به بهترین نحو بتوان عرصهٔ مسئولیت تضمین کیفیت را با شعار سازندگان اتومبیلی که زمانی مورد توجه همگان بود مشخص کرد: " حرف نخست را کیفیت می زند."

آنچه در مورد نرم‌افزار ضروری است، مسئولیتی است که تمام دست‌اندرکاران ساخت آن ـ مهندسان نرم‌افزار، مدیران پروژه، مشتریان، و افرادی که در درون یک گروه SQA کار می‌کنند ـ برای تضمین کیفیت نرم‌افزار باید در خود احساس کنند.

گروه SQA به‌عنوان نماینده دایم مشتری عمل می‌نماید. به این معنی، که اجراکنندگان SQA باید از نگاه مشتری به نرم‌افزار بنگرند ـ این‌که آیا نرم‌افزار به اندازهٔ کافی با فاکتورهای کیفیت هم‌خوانی دارد یا نه در فصل ۱۹ مورد اشاره قرار گرفته است. آیا توسعه نرم‌افزار بر طبق استانداردهای از قبل وضع شده اجرا شده است؟ آیا به‌عنوان بخشی از فعالیت SQA دیسیپلین فنی به‌خوبی نقش خود را ایفا کرده است؟ به‌منظور اطمینان از تضمین کیفیت نرم‌افزار گروه SQA تلاش می‌کند به این سئوال‌ها و نیز سئوال‌های دیگر پاسخ گوید.

۸-۳-۲ فعالیتهای تضمین کیفیت نرم افزار (SQA)

تضمین کیفیت نرم‌افزار از انواع گوناگونی از وظائف تشکیل یافته است که به دو گروه متفاوت مربوط می‌باشند ـ مهندسان نرم‌افزار که کار فنی انجام می‌دهند و یک گروه SQA که مسئولیت برنامه‌ریزی تضمین کیفیت، نظارت، ثبت و ضبط، تجزیه و تحلیل و گزارش را برعهده دارند.

مهندسان نرم‌افزار با اجرای روش‌های فنی و مقیاس‌های ثابت، اجرای بررسی‌های فنی منظم، و اجرای آزمون‌های نرم‌افزاری خوب برنامه‌ریزی شده، کیفیت را مورد توجه قرار می‌دهند (و تضمین کیفیت و فعالیت‌های کنترل کیفیت را عملی می‌کنند). در این فصل تنها بازبینی‌ها مورد بحث قرار گرفته‌اند. موضوعات فنی در بخش‌های ۳ تا ۵ این کتاب بحث شده‌اند.

نقش گروه تضمین کیفیت نرم افزار (SQA) کدام است؟

منشور گروه SQA تلاش دارد تا تیم نرم‌افزاری را در جهت رسیدن به کیفیت بالا در محصول نهایی ترغیب کند. مؤسسهٔ مهندسی نرم‌افزاری [PAU93] یک‌گروه از فعالیت‌های SQA را بیان می‌دارد که برنامه‌ریزی تضمین کیفیت، نظارت، ثبت، تجزیه و تحلیل و گزارش‌دهی را مورد توجه قرار می‌دهد. این فعالیت‌ها توسط یک گروه SQA مستقل انجام (یا تسهیل) می‌شوند:

یک طرح SQA برای پروژه آماده می‌کند. طرح در مدت برنامه‌ریزی پروژه به‌وجود می‌آید و توسط تمام گروه‌های علاقه‌مند بررسی می‌شود. فعالیت‌های تضمین کیفیت توسط تیم مهندسی نرم‌افزار اجرا می‌شود و گروه SQA به‌وسیله طرح، به پیش می‌رود. طرح به تشخیص موارد ذیل می‌پردازد:

- ارزشیابی‌هایی که باید انجام گیرد.

1.Schulmeyer,G.C.

- حسابرسی‌ها و بازبینی‌هایی که باید انجام گیرد.
- استانداردهایی که در مورد پروژه قابل اعمال‌اند.
- روال‌های گزارش و ردگیری خطاها.
- مستنداتی که باید توسط گروه SQA به‌وجود آیند.
- مقدار بازخوردی که به تیم پروژهٔ نرم‌افزاری بازگردانده شده است.

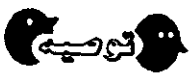
مشارکت در توصیف فرآیند نرم‌افزاری پروژه، تیم نرم‌افزاری روند اجرای کار را انتخاب می کند. گروه SQA توصیف فرآیند را برای سازگاری با سیاست سازمانی، استانداردهای داخلی نرم‌افزار، استانداردهای شدیداً الزامی (مثل ISO ۹۰۰۱)، و دیگر بخش‌های طرح پروژهٔ نرم‌افزاری بررسی می‌کند.

فعالیت‌های مهندسی نرم‌افزار را به‌منظور آگاهی از تطابق آنها با فرآیند تعریف شدهٔ نرم‌افزاری مورد بررسی قرار می‌دهد. گروه SQA به شناسایی مدارک می‌پردازد و انحرافات به‌وجود آمده در روند فرآیند را پی‌گیری می‌کند و نیز در مورد اصلاحات به وجودآمده به تحقیق می‌پردازد.

محصولات کاری طراحی‌شده نرم‌افزاری، وارسی می‌شود تا تطابق آنها با آنچه به‌عنوان بخشی از فرآیند نرم‌افزاری تعریف گردیده است مشخص شود. گروه SQA محصولات کاری منتخب را بررسی می‌کند، مدارک را شناسایی می‌کند؛ انحرافات را پی‌گیری می‌کند، در مورد اصلاحات به‌وجود آمده به تحقیق می‌پردازد، و متناوباً نتیجهٔ کار خود را به مدیر پروژه گزارش می‌دهد.

اطمینان حاصل می‌کند که انحرافات در کار نرم‌افزاری و محصولات کاری ثبت شده‌اند و بر اساس یک روال مستند با آنها برخورد می‌شود. انحرافات ممکن است در طرح پروژه، توصیف فرآیند، استانداردهای قابل اجرا، یا محصولات کاری فنی رخ بنمایند.

هر نوع ناهماهنگی را ثبت و به مدیریت عالی گزارش می‌دهد. موارد ناهماهنگی تا زمان بر طرف شدن آنها پی‌گیری می‌شوند. افزون بر این فعالیتها، گروه SQA کنترل و مدیریت تغییرات را، با هم هماهنگ می‌کند (فصل ۹) و به جمع‌آوری و تحلیل متریک‌های نرم‌افزاری کمک می‌کند.



مانند صافی های آب، شیوه های بازبینی رسمی جریان نرم افزار را تصفیه می کند. در این جریان فعالیتهای مهندسی، کندی به وجود می آورند. با استفاده از متریک ها تشخیص دهید که کدام بازبینی کار می کند و کدام کارآ نمی باشد. آنگاه بازبینی غیر موثر را از جریان خارج کنید.

۸-۴ بازبینی های نرم افزار

بررسی‌های نرم‌افزاری "فیلتری" برای فرآیند مهندسی نرم‌افزار می‌باشند. یعنی، در خلال توسعهٔ نرم‌افزار در نقاط مختلف بررسی‌ها اعمال می‌شوند و از آنها برای شناسایی خطاها و نقص‌ها و سپس بر طرف کردن آنها استفاده می‌شود. بررسی‌های نرم‌افزاری به‌منظور "پاک کردن" فعالیت‌های مهندسی نرم‌افزاری که ما آنها را تجزیه و تحلیل[1]، طراحی[1] و برنامه‌نویسی[2] می‌گوییم اعمال می‌شوند. فردمن و وین برگ [FRE90][3] لزوم بررسی‌ها را به این ترتیب مورد بحث قرار می‌دهند:

1 analysis

کار فنی به‌همان دلیلی که مدادها به پاک کن نیاز دارند، نیاز به بررسی دارد. اشتباه کردن[4] ویژگی آدمی[5] است. دومین دلیل لزوم وجود بررسی‌های فنی برای ما این است که اگر چه مردم در جلوگیری از بعضی از خطاهای خود خوب عمل می‌کنند، اما بسیاری از خطاها بسیار راحت‌تر از زیردست مسببین خود تا دیگران می‌گریزند. بنابراین روند بررسی پاسخی است به دعای روبرت برنز ***(Robert Burns)*** *:*

[خدایا] اندکی از قدرتی را به ما عنایت کن،تا خود را آن‌چنان که دیگران می‌بینند ببینیم.[6]

یک بررسی ـ از هر نوعی که باشد ـ راهی است برای به‌کار بردن گوناگونی گروهی از مردم برای:

۱. نشان دادن اصلاحات لازم در تولید یک فرد تنها یا یک گروه؛

۲. تأیید ان بخش‌هایی از تولید که یا علاقه‌ای به اصلاح آنها وجود ندارد یا اصولاً نیازی به اصلاح شدن ندارند؛

۳. دست‌یابی به کار فنی یک دست‌تر، یا دست کم، قابل پیش‌بینی‌تر به‌منظور بالابردن قابلیت مدیریت کار که در نتیجه آن می‌توان بدون انجام بررسی به کیفیت دست یافت.

انواع گوناگونی از بررسی‌ها وجود دارند که به‌عنوان بخشی از مهندسی نرم‌افزار قابل اجرا هستند و هر یک جایگاه خاص خود را دارند. اگر مسایل فنی مورد بحث قرار گیرند، یک جلسهٔ غیر رسمی در اطراف ماشین تهیهٔ قهوه نیز می‌تواند نوعی بررسی باشد. ارائه رسمی یک طرح نرم‌افزاری برای گروهی از مشتریان، مدیریت و کارکنان فنی نیز نوعی بررسی است. با این حال در این کتاب ما بر بررسی فنی رسمی **(FTR)** [7] که بعضی وقت‌ها وارسی[8] یا بازرسی[9] نیز گفته می‌شود تأکید داریم. از دیدگاه تضمین کیفیت یک بررسی فنی رسمی، مؤثرترین فیلتر می‌باشد. بررسی فنی رسمی که توسط مهندسان نرم‌افزار (و دیگران) برای مهندسان نرم‌افزار اجرا شده است، وسیله مؤثری برای بهبود بخشیدن کیفیت نرم‌افزاری است.

1.design

2.coding

3.freedman,D.P. and G.M.

4.err

5.human

6.O wad some power the giftie give us to see ourselves as other see us

7.Formal Technical Review

8.walkthrough

9.inspection

۸-۴-۱ تاثیر عیوب بر هزینه نرم افزاری

واژه‌نامهٔ اصطلاحات استاندارد برق و الکترونیک[1] یک نقص[2] را به‌عنوان یک "ناهنجاری محصول" تعریـــف می‌نماید. تعریف "عیب"[3] در بافت سخت‌افزاری را نیز در این واژه نامه می‌توان یافت ۱۹۹۰ – ۶۱۰/۱۲ :

الف) عیبی در یک وسیله یا مؤلفهٔ سخت‌افزاری؛ به‌عنوان مثال، یک مدار کوتاه یا سیم قطع شده.

ب) یک مرحله، فرآیند یا تعریف دادهٔ ناصحیح در یک برنامهٔ کامپیوتری، توجه: این تعریف در وحلهٔ نخست در دیسیپلین تحمل نقص به‌کار برده می‌شود. در کاربرد رایج، اصطلاحات "خطا" و "اشکال" برای بیان این معنی به‌کار می‌روند. همچنین نگاه کنید به: نقص ناشی از حساسیت داده‌ای؛ نقص ناشی از حساسیت برنامه‌ای؛ نقص قرینه؛ و نقص متناوب.

در بافت فرآیند نرم‌افزاری، اصطلاحات عیب و نقص مترداف هستند. هر دو متضمن یک اشکال کیفی هستند. که بعد از رسیدن نرم‌افزار به‌دست مصرف‌کننده پایانی کشف شده‌اند (یا به فعالیت دیگری در فرآیند نرم‌افزار).

در فصل‌های قبلی، ما اصطلاح "خطا"[4] را برای بیان یک اشکال کیفی به‌کار بردیم که به‌وسیلهٔ مهندسان نرم‌افزار (یا دیگران) قبل از این‌که نرم‌افزار به مصرف‌کنندهٔ پایانی (یا به فعالیت دیگری در فرآیند نرم‌افزار) عرضه گردد پیدا شده است.

هدف اصلی شیوه های بازبینی رسمی (FTR) یافتن خطا ها پیش از عبور از فعالیتی به فعالیت بعدی مهندسی نرم افزار (یا محصول را در اختیار مشتری نهادن) می باشد.

اولین هدف بررسی‌های فنی رسمی یافتن خطاها در جریان فرآیند است، به‌طوری‌که آنها بعد از عرضهٔ نرم‌افزار تبدیل به نقص نگردند. مزیت آشکار بررسی‌های فنی رسمی کشف زود هنگام خطاهاست، به‌طوری‌که آنها نتوانند به مرحلهٔ بعدی در فرآیند نرم‌افزار، نشر یابند. تعدادی از مطالعات صنعتی (از جمله **TRW, Nippon Electric, Mitre Corp**) نشان می‌دهند که فعالیت‌های طراحی در میان ۵۰ تا ۶۵ درصد خطاها (و در نهایت، همهٔ نقص‌ها) در خلال فرآیند نرم‌افزاری آغاز می‌گردند. با این‌حال، نشان داده شده است که فنون بررسی رسمی تا ۷۵ درصد در آشکار کردن نقایص طراحی مؤثر می‌باشند. [JON86][5] با شناسایی و رفع درصد بالایی از این خطاها فرآیند بررسی‌ها به‌طور عمده‌ای، هزینهٔ مراحل بعدی توسعه و تضمین را کاهش داده است.

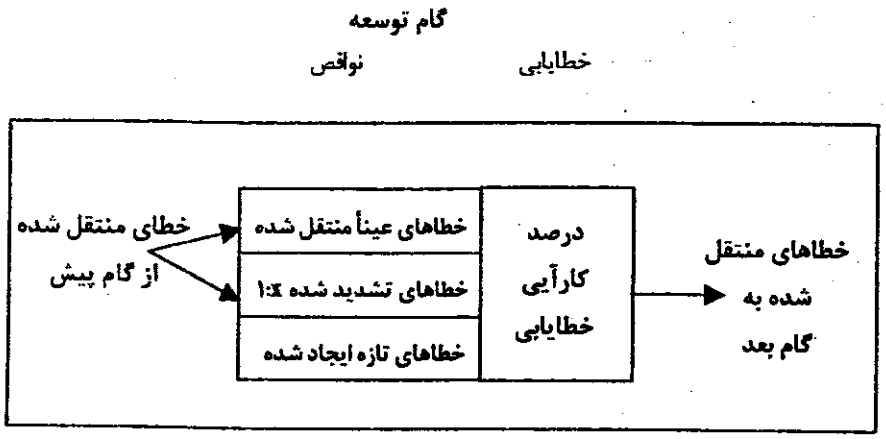
1.IEEE STANDARD dictionary of Electrical and electronics Terms

2.defect

3.fault

4.error

5.Jones, T.C.

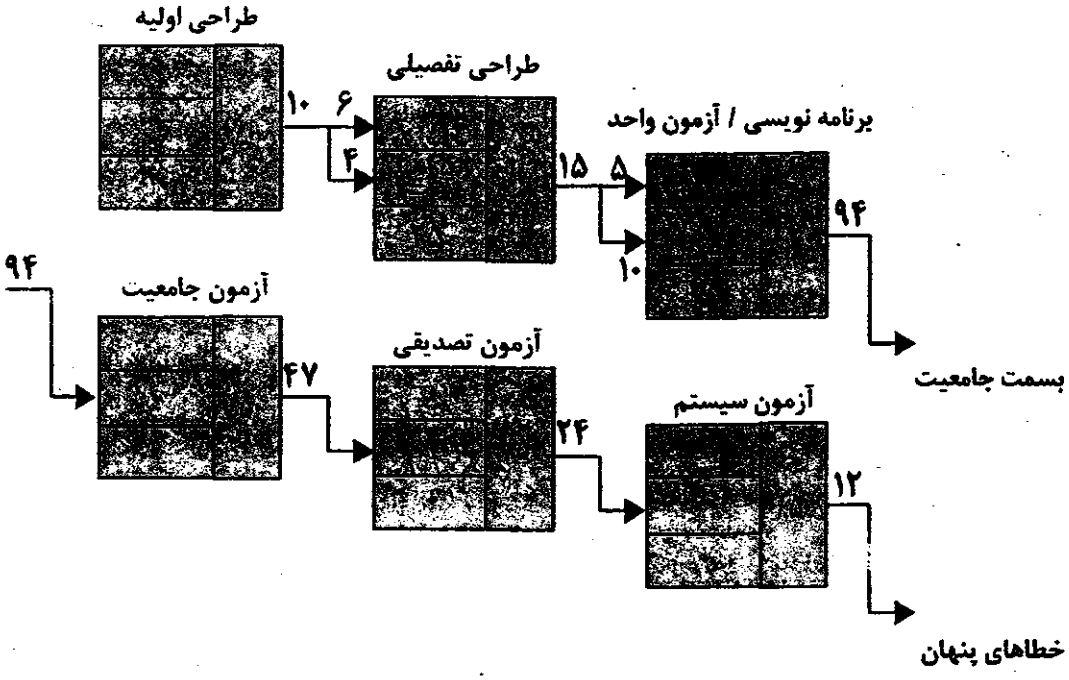


شکل ۸-۲ مدل تشدید و توسعه نقص

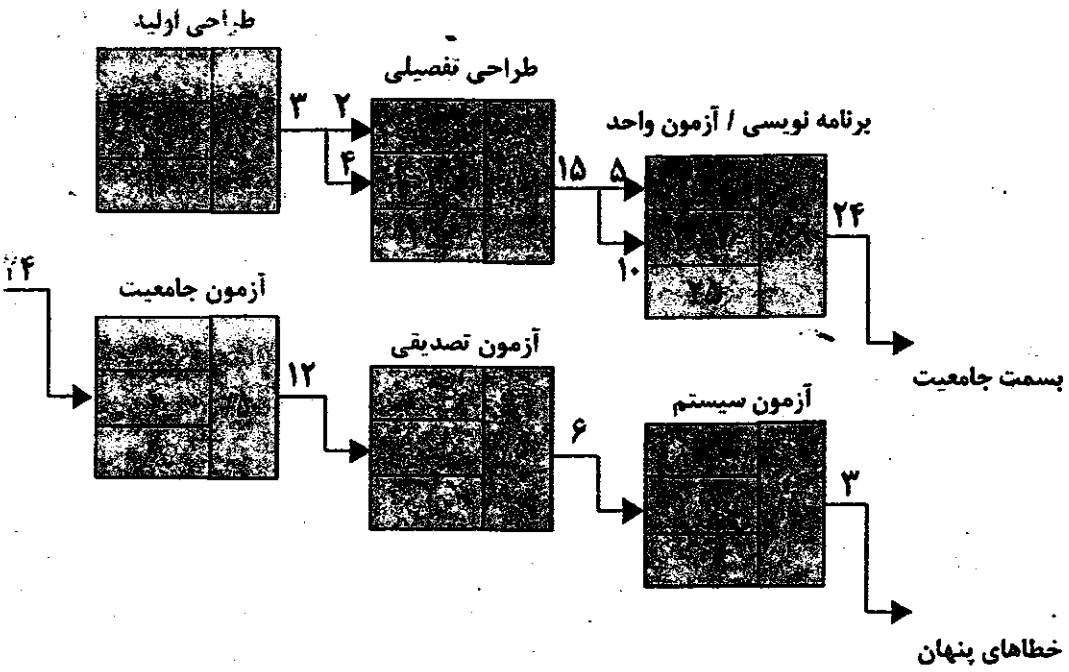
برای نشان دادن اثر هزینهٔ ردیابی زود هنگام خطاها، ما یکسری از هزینه‌های نسبی‌ای را که بر اساس داده‌های واقعی هزینه‌ها برای پروژه‌های نرم‌افزاری بزرگ جمع‌آوری شده‌اند مورد ملاحظه قرار می‌دهیم. [IBM81][۱].[۲] فرض کنید که اگر خطایی در جریان طراحی آشکار گردد، برای برطرف کردن آن ۱/۰ واحد مالی هزینه لازم است. در همین رابطه هزینه در خلال مدت آزمون ۱۵ واحد مالی و بعد از عرضهٔ کالا بین ۶۰ تا ۱۰۰ واحد مالی خواهد بود.

1.Implementing Software Inspections

۲. هرچند این داده ها، ۲۰ سال قدمت دارند، هنوز در متن های مدرن و جدید قابل کاربرد می باشند.



شکل ۸-۳ تشدید نقص و عیب بدون بازبینی



شکل ۸-۴ تشدید نقص و عیب با بازبینی

۸-۴-۲ تشدید نقص و برطرف کردن آن

یک مدل توسعهٔ نقص برای تشریح تولید و ردیابی خطاها در جریان طراحی اولیه، طراحی تفصیلی و مراحل برنامه‌نویسی فرآیند مهندسی نرم‌افزار قابل کاربرد است. این مدل به‌صورت نمودار در تصویر ۸-۲ نشان داده شده است. هر جعبه یکی از مراحل توسعه نرم‌افزار را نشان می‌دهد. در جریان هر مرحله، خطاها ممکن است سهواً تولید شوند. بررسی‌ها ممکن است نتوانند خطاهای جدید و یا خطاهای قبلی را آشکار کنند، و در نتیجه تعدادی خطا را دوباره فعال کند. در بعضی موارد خطاهای باقی‌مانده از مراحل قبلی به‌وسیله کار کنونی گسترش می‌یابند (عامل تشدید،×). تقسیمات فرعی جعبه‌ها نمایانگر هر یک از این ویژگی‌ها و درصد کارآیی خطاهای ردیابی شده می‌باشند، که کارکردی از اجرای کامل بررسی‌ها می‌باشد.

تصویر ۸-۳ یک مثال فرضی از توسعه و تشدید نقص در یک فرآیند توسعهٔ نرم‌افزاری را نشان می‌دهد که در آن هیچ‌گونه بررسی انجام نشده است. با توجه به تصویر، فرض می‌شود که هر یک از مراحل آزمون برای آشکارسازی و تصحیح ۵۰ درصد از همهٔ خطاهای کنونی، بدون تولید خطاهای جدید، به‌کار می‌رود (فرضی خوش‌بینانه). قبل از آغاز مراحل آزمون ده نقص طراحی اولیه به ۹۴ خطا گسترش یافته و تشدید شده‌اند. دوازده خطای پنهان نیز وارد محیط کار شده‌اند. تصویر۸-۴ نیز شرایط مشابهی را مورد ملاحظه قرار می‌دهد، با این استثنا که بررسی طراحی و برنامه به‌عنوان بخشی از هر مرحله انجام گرفته است. در این مورد، ده خطای طراحی اولیه قبل از آغاز آزمون‌ها، به ۲۴ خطا افزایش یافته‌اند ـ تنها سه خطای پنهان وجود دارد. با به یاد آوردن هزینه‌های نسبی مربوط به کشف و تصحیح خطاها، ارزش کلی قابل بیان خواهد بود (باو بدون بررسی مثال فرضی‌مان). تعداد خطاهای آشکار شده در هر یک از مراحل مورد اشارهٔ تصویر ۸-۳ و ۸-۴ با هزینهٔ برطرف کردن هر خطا افزایش یافته است. (هزینه در طراحی ۱/۵ واحد، قبل از آزمون ۶/۵ واحد هزینه، در جریان آزمون ۱۵ واحد هزینه و بعد از عرضهٔ محصول ۶۷ واحد هزینه می‌باشد)با به‌کار بردن این داده‌ها، هزینهٔ کلی توسعه و نگهداری در جریان انجام بررسی‌ها برابر است با ۷۸۳ واحد هزینه. اگر هیچ‌گونه بررسی‌ای انجام نگرفته باشد، هزینهٔ کلی برابر خواهد بود با ۲۱۷۷ واحد هزینه ـ یعنی تقریباً با سه برابر هزینهٔ قبلی.

برای انجام بررسی‌ها، یک مهندس نرم‌افزار باید زمان و تلاش صرف کند و سازمان توسعه‌دهنده نیز باید پول خرج نماید. با این‌حال، نتایج مثال قبلی جای کمترین شکی باقی نمی‌گذارد که ما با سیندرمی (Syndrome) مواجهیم که می‌گویند "یا اکنون پرداخت کن یا بعداً بیشتر پرداخت کن." بررسی‌های فنی منظم (برای طراحی و دیگر فعالیت‌های فنی) سود قابل ملاحظه‌ای به‌بار می‌آورد، آنها باید انجام گیرند.

نقل قول

برخی بیماریها، همنطور که پزشکان می گویند، در ابتدا به راحتی درمان می شود اما تشخیص آنها مشکل است... اما با گذشت زمان تشخیص آنها با عوارضشان بسیار ساده خواهد شد و البته درمان بسیار مشکل خواهد بود.
نیکولو ماکیاولی

نقل قول

چه هنگام و با چه هدفی بازبینی های رسمی را بکار می بریم؟

۸-۵ بازبینی های فنی رسمی

یک بررسی فنی رسمی (FTR) یک فعالیت تضمین کیفیت نرم‌افزار است که توسط مهندسان نرم‌افزار (و دیگران) اجرا می‌شود. اهداف FTR عبارتند از: (۱) آشکار کردن خطاهای موجود در کارکرد،

منطق یا اجرای هر نوع ارائه نرم‌افزار؛ (۲) بررسی این نکته که نرم‌افزار تحت بررسی با نیازمندیهایش هم‌خوانی دارد؛ (۳) اطمینان از این‌که نرم‌افزار بر اساس استانداردهای از پیش تعریف شده ارائه شده است؛ (۴) دستیابی به نرم‌افزاری که به شیوه‌ای همگن تولید شده باشد، و (۵) افزایش قابلیت ادارهٔ پروژه‌ها. افزون بر این، FTR به‌عنوان یک زمینهٔ آموزشی به‌کار می‌رود، و مهندسان جوان را قادر می‌سازد که رهیافتهای متفاوت تجزیه و تحلیل نرم‌افزاری، طراحی و اجرا را مشاهده کنند. FTR همچنین موجب تقویت پیشرفت و انسجام کار می‌شود، زیرا باعث می‌شود گروهی از افراد به این وسیله با بخش‌های نرم‌افزار آشنا گردند که در غیر این‌صورت امکان آشنایی با آنها برایشان به‌وجود نمی‌آمد.

FTR در واقع گروه‌ای از بررسی‌هاست که شامل، جستجوها، بازرسی‌ها، بررسی‌های مخفی و دیگر ارزیابی‌های فنی گروه‌های کوچک نرم‌افزاری است. هر FTR به‌عنوان یک گردهم‌آیی اجرا می‌گردد و تنها وقتی مؤفق خواهد بود که به‌طور مناسب برنامه‌ریزی، کنترل و نظارت شده باشد. در بخش‌هایی که به‌دنبال خواهد آمد، راهنمایی‌هایی شبیه آنچه برای جستجوها مطرح گردید [GIL93][1] و [FRE90][2] به‌عنوان یک نمونهٔ بررسی فنی رسمی ارائه شده‌اند.

۸-۵-۱ نشست بازبینی

صرف‌نظر از قالبی که برای FTR انتخاب شده است، هر گردهم‌آیی بررسی باید به‌وسیلهٔ قیود ذیل به پیش رود:

- معمولاً سه تا ۵ نفر باید در این بررسی دخالت داشته باشند.
- آمادگی قبلی باید وجود داشته باشد، اما نباید برای هر فرد بیش از دو ساعت وقت نیاز داشته باشد.
- مدت زمانی گردهم‌آیی بررسی باید کمتر از دو ساعت باشد.

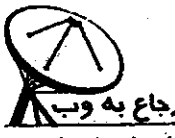
با در نظر داشتن قیود فوق، روشن است که FTR بر روی بخش خاص (و کوچکی) از نرم‌افزار کلی تمرکز می‌کند. به‌عنوان مثال، به‌جای تلاش برای بررسی کل یک طراحی، جستجوها در مورد هر جزء یا گروه کوچکی از اجزاء اجرا می‌شوند. با محدود شدن تمرکز، احتمال خطاهای آشکار نشدهٔ بیشتر FTR توسعه می‌یابد.

تمرکز FTR بر روی یک محصول کاری است (مثلاً نسبتی از یکی از مشخصه‌های نیازمندیها، یک طراحی مبتنی بر جزء، تهیه لیستی برای کد مبنای یک جزء). فردی که محصول کاری را به انجام رسانده است ـ تولیدکننده[3] ـ رهبر پروژه را از اتمام محصول کاری و نیاز به انجام یک بررسی آگاه می‌سازد. رهبر

نقل قول

نشستها اغلب برای دقایقی مفید است، و ساعتها اتلاف وقت.
ناشناس

یک بازبینی رسمی، متمرکز بر بخش کوچک مرتبطی از یک محصول کاری است.



ارجاع به وب

کتاب راهنمای بازرسی رسمی ناسا (NASA SATC) از آدرس زیر قابل پیاده سازی است:
www.satc.gsfc.nasa.gov/fi/fipage.html

1.Freedman, D.P. and G.M.

2.Gilb,T.and D.

3.producer

پروژه با یک متخصص بررسی[1] تماس می‌گیرد که او تولید را از لحاظ آمادگی ارزشیابی می‌کند، و نمونه‌هایی از مواد حصول را تولید و برای آماده‌سازی بهتر در اختیار دو یا سه بررسی‌کننده قرار می‌دهد. انتظار می‌رود که هر بررسی‌کننده یک تا دو ساعت برای بررسی محصول وقت صرف کنند و بعد دربارهٔ آن یادداشت‌هایی تهیه کنند و در غیر این‌صورت با کار آشنا گردند. همزمان، مسئول بررسی نیز محصول را بررسی می‌کند و دستور کاری برای گردهم‌آیی بررسی تهیه می‌کند، که معمولاً برای روز بعد زمان‌بندی شده است.

گردهم‌آیی بررسی با شرکت مسئول بررسی، همهٔ بررسی‌کنندگان و تولیدکننده تشکیل می‌گردد. یکی از بررسی‌کنندگان نقش ثبت‌کننده[2] را به‌عهده می‌گیرد، که فردی است که تمام موضوعات مهمی را که در جریان بررسی به‌وجود می‌آید (به‌صورت کتبی) ثبت می‌کند. (FTR) با مقدمه‌ای از دستور کار و نیز مقدمهٔ مختصری توسط تولیدکننده آغاز می‌گردد. سپس تولیدکننده به جریان تولیدکار اشاره می‌کند، و زمانی‌که بررسی‌کنندگان مسایلی را برپایهٔ آماده‌سازی بهتر مطرح می‌کنند، به تبیین موارد می‌پردازد. هرگاه مشکلات یا خطاهای اساسی آشکار می‌گردند ثبت‌کننده آنها را ثبت می‌کند.

در پایان بررسی، تمام شرکت‌کنندگان FTR یک نوشتهٔ پایانی تهیه می‌کنند، و مشارکت و موافقت خود با یافته‌های تیم بررسی را نشان می‌دهند.

۸-۵-۲ گزارش بازبینی ها و ثبت موضوعات

در مدت کار FTR، یکی از بررسی‌کنندگان (فرد ثبت‌کننده) فعالانه تمام موضوعات مطرح شده را ثبت می‌کنند. این موضوعات ثبت شده در پایان گردهم‌آیی بررسی خلاصه می‌شوند و "لیستی از موضوعات بررسی شده" تهیه می‌گردد. همچنین، گزارش مختصری از بررسی فنی[3] رسمی امور تهیه می‌گردد. گزارش مختصر بررسی به سه سؤال پاسخ می‌دهد:

۱. چه چیزی بررسی شده است؟

۲. چه کسی آن را بررسی کرده است؟

۳. نتایج و یافته‌ها چه چیزهایی بودند؟

گزارش مختصر بررسی، یک فرم یک صفحه‌ای است (به‌همراه ملحقات احتمالی). این گزارش تبدیل به بخشی از مواد ثبت شدهٔ پروژه می‌شود و ممکن است به رهبر پروژه و دیگر گروه‌های علاقه‌مند ارائه گردد.

1.review leader

2.recorder

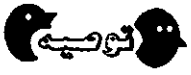
3.review summary report

لیست موضوعات[1] بررسی شده به دو منظور به‌کار می‌رود: (۱) شناسایی زمینه‌های مشکل‌دار محصول و (۲) به‌عنوان لیستی برای چک کردن عملکردها که تولیدکننده را به هنگام انجام اصلاحات راهنمایی می‌کند. لیست موضوعات معمولاً ضمیمهٔ گزارش مختصر می‌شود. همچنین ایجاد یک روال پی‌گیری برای اطمینان از این‌که موارد مطرح در لیست موضوعات بررسی شده به‌خوبی اصلاح گردیده‌اند دارای اهمیت است. در صورت عدم انجام این کار، این احتمال وجود دارد که موضوعات مطرح شده "در لابه‌لای امور دیگر گم شوند." واگذار کردن مسئولیت روال گفته شده به مسئول بررسی، یک رهیافت موجود می‌باشد.

۸-۵-۳ رهنمودهای بازبینی

لیست محتویات و خلاصه گزارش بازرسی فنی

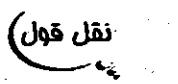
رهنمودهای اجرای بررسی‌های فنی رسمی باید از قبل تنظیم شده باشند، به آگاهی تمام بررسی‌کنندگان رسیده باشند، بر سر آنها توافق شده باشد و سپس بر حسب آنها عمل شود. یک بررسی کنترل نشده اغلب می‌تواند بدتر از نبودن هیچ‌گونه بررسی‌ای باشد. آن‌چه در پی می‌آید نمایندهٔ حداقلی از رهنمودها برای بررسی‌های فنی منظم است:



خطاها را به صورت خشن مورد اشاره قرار ندهید. یک راه مناسب، آن است که با طرح پرسشی، تولید کننده را به کشف خطای خویش رهنمون سازید.

۱- محصول را بررسی کنید[1] نه تولیدکننده را. یک **FTR** مردم و افراد را وارد جریان کار می‌کند. اگر به‌طور مناسب اجرا گردد، **FTR** در تمام مشارکت‌کنندگان احساس گرم اتمام کار را به‌وجود خواهد آورد. و اگر به‌طور نامناسب اجرا گردد، **FTR** شکل و بوی استنطاق و تفتیش عقاید را به خود خواهد گرفت. خطاها باید به آرامی ذکر گردند، لحن صحبت در جلسه باید متین و سازنده باشد؛ و قصد رنجاندن یا تحقیر کسی نباید در کار باشد. رهبر بررسی باید جلسه را هدایت کند و اطمینان حاصل کند که لحن و رفتار مناسب بر جلسه حکم‌فرماست و باید فوراً بررسی‌ای را که از کنترل خارج شده است متوقف کند.

۲- یک دستور کار، تنظیم و آن را حفظ کنید. یکی از اساسی‌ترین نقص‌های گردهم‌آیی‌ها از هر نوعی که باشند بی‌برنامگی است. یک **FTR** باید در مسیر خود و مطابق زمان‌بندی جلو رود. رهبر بررسی دارای مسئولیت حفظ و رعایت زمان‌بندی گردهم‌آیی است، و هرگاه بی‌برنامگی وارد کار شود نباید از تذکر دادن به افراد بیمی داشته باشد.



یک مفهوم زیبای زندگی این است که به کسی نمی توان بدون کمک و یاری خودش، یاری رساند.
والف والدو امرسون

۳- منازعه و جدل متقابل را محدود کنید. وقتی موضوعی توسط یک بررسی‌کننده مطرح می‌شود، ممکن است توافق همگانی بر روی اثر آن وجود نداشته باشد. به‌جای صرف زمان برای بحث دربارهٔ آن سؤال، موضوع باید برای بحث خارج از جلسه ثبت گردد.

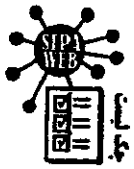
۴- زمینه‌های دارای مشکل را اعلام دارید، اما سعی نکنید تمام مشکلات اعلام شده را حل کنید. یک بررسی، جلسهٔ حل مسئله نیست. اغلب راه‌حل مسئله توسط خود تولیدکننده به

---

1 review issues

تنهایی یا با کمک فرد دیگری قابل یافتن است. حل مسئله باید تا گردهم‌آیی بررسی بعدی، به تعویق انداخته شود.

۵- **یادداشت‌های کتبی تهیه کنید.** بعضی وقت‌ها نوشتن یادداشت‌ها به‌وسیله ثبت‌کننده بر روی یک تابلوی دیواری می‌تواند عقیدهٔ خوبی باشد، به‌طوری‌که شیوهٔ گزینش واژه‌ها و تقدمات رعایت شده هم‌زمان با ثبت اطلاعات، به‌وسیلهٔ بررسی‌کنندگان دیگر قابل ارزیابی است.



چک لیست های FTR (بازبینی ها)

۶- **تعداد شرکت‌کنندگان را محدود کنید و بر روی آمادگی قبلی پافشاری کنید.** دو رییس از یکی بهتر است، اما ۱۴ رییس لزوماً از ۴ رییس بهتر نمی‌باشند. تعداد افراد شرکت‌کننده را در حداقل لازم نگه دارید. با این همه، همهٔ اعضای تیم بررسی از پیش باید آماده باشند. توصیه‌های مکتوب باید توسط مسئول بررسی انتخاب گردند (که می‌تواند نشان‌دهندهٔ این نکته باشد که بررسی‌کننده همهٔ مواد را بررسی کرده است).

۷- **لیستی برای هر محصول که احتمال بررسی آن می‌رود تهیه کنید.** این لیست به مسئول بررسی کمک می‌کند تا ساختار گردهم‌آیی **FTR** را تعیین کند و به هر بررسی‌کننده کمک می‌کند تا بر موضوعات مهم تمرکز کند. این لیست باید به‌منظور تجزیه و تحلیل، طراحی، برنامه‌نویسی و حتی آزمون مستندات به‌کار رود.

۸- **منابع و زمان لازم را به FTR اختصاص دهید.** برای کارآمد بودن بررسی‌ها، آنها را باید به‌عنوان یک کار در مدت روند مهندسی نرم‌افزار در نظر گرفت. همچنین زمان لازم برای تغییرات غیرقابل اجتنابی که در نتیجهٔ یک **FTR** اتفاق می‌افتند باید در نظر گرفته شود.

۹- **آموزش‌های هدف‌مندی برای همهٔ بررسی‌کنندگان فراهم کنید.** به‌منظور کارآمد بودن شرکت‌کنندگان بررسی باید یک‌سوی آموزش‌های رسمی دریافت دارند. آموزش‌ها باید بر هر دو جنبهٔ موضوعات مربوط به روند کار و جنبهٔ روانی انسانی بررسی‌ها تکیه داشته باشد. فردمن و وینبرگ **[FRE90]**[2] یک منحنی یادگیری یک ماهه برای ۲۰ نفر را که به‌صورت مؤثر در بررسی‌ها شرکت کردند مورد ارزیابی قرار دادند.

۱۰- **بررسی‌های اولیهٔ خود را بررسی کنید.** پرسیدن می‌تواند در آشکار کردن مشکلات در خود روند بررسی سودمند باشد. اولین محصولات مورد بررسی ممکن است خود تبدیل به رهنمودهای بررسی‌ها شوند.

از آن‌جا که متغیرهای بسیاری وجود دارند (به‌عنوان مثال، تعداد شرکت‌کنندگان، نوع محصول‌کاری، زمان‌بندی و طول آن، رهیافت بررسی خاص) که بر یک بررسی مؤفق تأثیر می‌گذارند، یک سازمان

1.Review the product

2.Freedman, D.P.

نرم‌افزاری برای تعیین رهیافتی که بهترین کارکرد را در یک بافت محلی دارد باید به آزمون بپردازد. پورتر [POR95][1] و همکارانش راهنمایی‌های خوبی برای این نوع آزمون‌ها به‌دست می‌دهند.

۸-۶ رهیافت های رسمی برای تضمین کیفیت نرم افزار

فنونی برای تعیین مشخصات رسمی نرم افزار درفصل ۲۵ ارائه گردیده اند. تعیین صحت ها نیز در فصل ۲۶ قرار گرفته اند.

در قسمت های پیشین به این اعتقاد رسیدیم که کیفیت نرم افزار بر عهده همه افراد است، که در مراحل تحلیل و طراحی و برنامه سازی و آزمون همانقدر باید مورد توجه باشد که در بازبینی های رسمی فنی، در راهبرد های تلاش چندگانه آزمون، و کنترل بهتر محصولات کاری و تغییر آنها و دریافت استاندارد های مهندسی نرم افزار. بعلاوه، کیفت می تواند به طیف گسترده ای از عوامل کیفی و اندازه های غیر مستقیم تعبیر شود که شاخص ها و متریک های متعدد آن را نشان می دهند.

در دو دهه گذشته، یک بخش کوچک اما ویژه از جامعه مهندسی نرم افزار به این اعتقاد رسیدند که یک رهیافت با رسمیت بیشتر برای تضمین کیفیت نرم افزار لازم است. آنان معتقد بودند (و هستند) که یک برنامه کامپیوتری یک شئ ریاضی محسوب می شود.[SOM96]. یک نحو و معنای سخت و شدید می تواند در هر زبان برنامه سازی تعریف شود که در اینصورت رهیافت سختی نیز برای مشخصه های نیازمندیهای نرم افزاری در پیش رو خواهد بود. اگر مدل نیازمندیها (مشخصه ها) و زبان برنامه سازی بتوانند به صورتی محکم سخت و مطمئن بیان شوند می توان تعیین صحت ریاضی نیز برای آنها داشت تا به این تضمین برسیم که مشخصه های آنها کاملا" پوشش داده شده است.

برای تضمین کیفیت نرم افزار به طور آماری چه کارهایی لازم است؟

تلاش برای تضمین و تعیین صحت برنامه ها امر جدیدی نیست. دایکسترا [DIJ76] و لینجر، میلز و ویت [LIN79] و بسیاری دیگر به تعیین صحت و تضمین برنامه ها پرداخته اند و این امر منجر به استفاده از مفاهیم برنامه سازی ساختیافته گردیده است.

۸-۷ تضمین کیفیت آماری نرم افزار

۸۰ درصد عیبها در ۲۰ درصد از برنامه واقع شده است. آنها را بیابید و رفع کنید.
لاول آرتور

تضمین کیفیت آماری[2] منعکس‌کنندهٔ یک روند رو به رشد در صنعت در جهت کمّی شدن بیشتر نسبت به کیفیت است. در مورد نرم‌افزار تضمین کیفیت آماری متضمن مراحل زیر است:

۱- اطلاعات دربارهٔ نقص‌های نرم‌افزار جمع‌آوری و مقوله‌بندی می‌شود.

۲- تلاشی برای ردیابی علت زیربنایی هر نقص انجام می‌گیرد (عدم هماهنگی با مشخصات، خطای طراحی، انحراف از استانداردها، ارتباط ضعیف با مشتری).

1.Porter, A.

2.Statestical quality assurance

۳- اصول پارتو به کار بسته می شود(۸۰٪ نواقص در ۲۰٪ علل، ریشه دارند) ، و آن ۲۰٪ یافت می شود.(علل حیاتی)

۴- در حین شناسایی علل حیاتی (مرحله قبل) حرکت به سمت مسائلی که ریشه نواقص بوده اند انجام می گیرد.

به‌منظور کاربرد **SQA** آماری جدول ۸-۱ طراحی شده است. این جدول نشان می‌دهد که , **MCC EDR, IES** به‌عنوان دلایل اساسی[1] انتخاب خواهند شد در حالیکه علت ۵۳ درصد تمام خطاها هستند. اگر فقط خطاهای اساسی موردنظر باشند، **MCC , EDR, IES** به عنوان دلایل اساسی انتخاب خواهند شد. بعد از این‌که دلایل اساسی تعیین گردیدند، سازمان مهندسی نرم‌افزاری، ممکن است برای بهبود کیفیت ارتباط مشتری و مشخصات تعیین شده، به اجرای فنون مشخصات کاربرد تسهیل شده بپردازد (فصل ۱۱). برای بهبود **EDR** سازنده ممکن است برای مدل‌سازی داده‌ها از ابزارهای **CASE** استفاده کند و به اجرای بررسی‌های طراحی داده‌ای شدیدتری دست بزند.

	کل		مهم		متوسط		جزئی	
نوع خطا	تعداد	درصد	تعداد	درصد	تعداد	درصد	تعداد	درصد
IES	205	22%	34	27%	68	18%	103	24%
MCC	156	17%	12	9%	68	18%	76	17%
IDS	48	5%	1	1%	24	6%	23	5%
VPS	25	3%	0	0%	15	4%	10	2%
EDR	130	14%	26	20%	68	18%	36	8%
ICI	58	6%	9	7%	18	5%	31	7%
EDL	45	5%	14	11%	12	3%	19	4%
IET	95	10%	12	9%	35	9%	48	11%
IID	36	4%	2	2%	20	5%	14	3%
PLT	60	6%	15	12%	19	5%	26	6%
HCI	28	3%	3	2%	17	4%	8	2%
MIS	56	6%	0	0%	15	4%	41	9%
جمع کل	942	100%	128	100%	379	100%	435	100%

جدول ۸ـ۱ مجموعه داده‌های آزمون تضمین کیفیت آماری نرم‌افزار (SQA)

ذکر این نکته دارای اهمیت است که فعالیت اصلاحی در وهلهٔ نخست بر علل اساسی متمرکز است ـ با تصحیح علل اساسی، مسایل جدیدی برای تصحیح شدن پیدا می‌شود.

1.vital few

فنون تضمین کیفیت آماری برای نرم‌افزار، نشان‌دهندهٔ فراهم آوردن بهبود کیفیت آماری است [ART97][1]. در بعضی موارد، سازمان‌های نرم‌افزاری، بعد از به‌کار بستن این تکنیک‌ها به یک کاهش ۵۰ درصدی در نقایص در طول یک‌سال دست یافته‌اند.

در کنار جمع‌آوری اطلاعات دربارهٔ نقایص کار، سازندگان نرم‌افزار می‌توانند یک شاخص خطا (EI) برای هر یک از مراحل اصلی فرآیند نرم‌افزار محاسبه کنند [IEE94]. بعد از تجزیه و تحلیل، طراحی، برنامه‌نویسی، آزمودن و عرضه، اطلاعات زیر جمع‌آوری شده‌اند:

E_i = تعداد کل خطاهایی که در خلال مرحلهٔ i ام روند مهندسی نرم‌افزار آشکار شدند.

S_i = تعداد خطاهای اساسی

M_i = تعداد خطاهای میانه

T_i = تعداد خطاهای کوچک

PS = اندازه محصول (LOC، بیان طراحی، صفحات مستند) در مرحله i ام

W_t , W_m , W_s = فاکتورهای وزنی خطاهای اساسی، میانه و کوچک، در جایی‌که مقدارهای داده شده عبارتند از $W_s = ۱۰$ ، $W_m = ۳$، $W_t = ۱$. با پیشرفت و توسعه کار عامل‌های وزنی هر مرحله باید بزرگ‌تر گردند. این به نفع سازمانی است که خطاها را در مراحل اولیه پیدا می کند.

در هر مرحله از فرآیند نرم‌افزاری، یک شاخص مرحله، PI_i ، محاسبه می‌شود:

$$PIi = W_S (S_i / E_i) + W_m (M_i / E_i) + W_t (T_i / E_i)$$

شاخص خطا (E_i) با محاسبهٔ اثر انباشتنی هر PI_i محاسبه شده است. خطاهای وزنی بعداً در روند مهندسی نرم‌افزار به‌صورت سنگین‌تری نسبت به مراحل قبل ظاهر می‌شوند.

PI1.)Ips*EI= (I

$$EI = \sum(i \times PI_i) / PS$$

$$= (PI_1 + PI_2 + PI_3 + ...+ i\,PIi) / PS$$

برای توسعهٔ یک نشان کلی از بهبود به‌وجود آمده در کیفیت نرم‌افزار می‌توان شاخص خطا را به همراه اطلاعات گردآوری شده در جدول ۸-۱ به‌کار برد.

کاربرد SQA آماری و قاعدهٔ پرتو را در یک جملهٔ واحد می‌توان خلاصه کرد: زمان خود را صرف تمرکز بر چیزهایی کنید که حائز اهمیت هستند، اما نخست مطمئن شوید که شما واقعاً درک می‌کنید چه چیزی مهم است!

---

1 Arthure, L.J.

بحث جامع SQA آماری از حوصلهٔ این کتاب خارج است. خوانندگان علاقه‌مند باید به [SCH98][1]، [KAP95][2] یا [KAN95][3] مراجعه کنند.

ارجاع به وب

مرکز تحلیل قابلیت اطمینان، اطلاعات بسیار مفیدی را درخصوص قابلیت اطمینان، نگهداری، پشتیبانی، و کیفیت در آدرس زیر قرارداده است :
www.raciitri.org

۸-۸ قابلیت اطمینان نرم افزار

بی‌هیچ شکی قابلیت اطمینان یک برنامهٔ کامپیوتری عنصر مهمی در کیفیت کلی نرم‌افزار می‌باشد. اگر یک برنامهٔ کامپیوتری مرتباً و به‌طور متوالی دارای مشکل اجرایی گردد، تفاوت چندانی ندارد که آیا دیگر فاکتورهای کیفیت نرم‌افزار قابل قبول‌اند یا نه.

اعتبار نرم‌افزار، برخلاف بسیاری دیگر از فاکتورهای کیفیت، با به‌کار بردن اطلاعات تاریخی و توسعه‌ای به‌طور مستقیم قابل اندازه‌گیری است و می‌توان آن را برآورد کرد. در اصطلاحات آمار اعتبار نرم‌افزاری به‌صورت "احتمال خرابی خارج از کاربرد یک برنامهٔ کامپیوتری در یک محیط خاص و در یک زمان مشخص." [MUS87][4] برای روشن شدن موضوع، برنامه x برآورد شده است که دارای اعتباری به میزان ۰/۹۶ است در طول ۸ ساعت فرآیند سپری شده. به‌عبارت دیگر، اگر برنامهٔ x برای صدبار به اجرا درآمدن طراحی شده باشد و نیاز به ۸ ساعت زمان فرآیند سپری شده داشته باشد (زمان اجرا)، احتمال دارد که ۹۶ بار از ۱۰۰ بار مورد انتظار را به‌طور صحیح عمل نماید.

هرگاه اعتبار نرم‌افزار مورد بحث قرار می‌گیرد، یک سؤال محوری مطرح می‌شود: منظور از اصطلاح خرابی چیست؟ دریافت هر بحثی دربارهٔ کیفیت و اعتبار نرم‌افزاری، خرابی یعنی ناسازگاری با نیازمندیهای نرم‌افزار. با این‌حال، حتی در درون این تعریف نیز درجه‌بندی موجود است، خرابی‌ها می‌توانند تنها اذیت‌کننده یا مصیبت‌بار باشند. یک خرابی را می‌توان در چند ثانیه تصحیح کرد در حالی‌که خرابی دیگری نیاز به هفته‌ها یا حتی ماه‌ها برای تصحیح شدن دارد. یک موضوع باعث پیچیدگی بیشتر کار می‌شود، تصحیح یک خرابی ممکن است منجر به آغاز خطاهای دیگری گردد که نهایتاً به خرابی‌های دیگر می‌انجامند.

مسائل مربوط به قابلیت اطمینان نرم افزار همگی غالباً" با ردگیری خطاهای طراحی یا پیاده سازی ، قابل رفع است.

۸-۸-۱ اندازه گیری قابلیت اطمینان و قابلیت دسترسی

کارهای اولیه بر روی اعتبار نرم‌افزاری تلاش داشتند تا ریاضیات تئوری اعتبار سخت‌افزاری را (مثلاً [ALV64][1]) برای پیش‌بینی اعتبار نرم‌افزاری به‌کار برند. بیشتر مدل‌های مربوط به اعتبار سخت‌افزاری بر مبنای پیش‌بینی خرابی ناشی از کهنگی و فرسودگی قرار دارند و نه برمبنای خرابی ناشی از نقایص طراحی

1.Schmauch
2.Kaplan, C.R.
3.Kan, S.H.
4.Musa,J.D.

در سخت‌افزار، خرابی‌های ناشی از کهنگی و فرسودگی فیزیکی (مثلاً، اثرات دما، پوسیدگی و ضربه) بسیار محتمل‌تر از خطاهای مربوط به طراحی است. متأسفانه، خلاف این موضوع در مورد نرم‌افزار صدق می‌کند. در واقع، تمام خرابی‌های نرم‌افزاری را می‌توان به مشکلات طراحی یا پیاده‌سازی مربوط دانست؛ کهنه‌گی (نگاه کنید به فصل ۱) در خرابی نرم‌افزاری نقش چندانی ندارد.

هنوز بحث بر سر رابطهٔ بین مفاهیم اصلی در قابلیت اطمینان سخت‌افزاری و قابلیت اجرایی آنها در نرم‌افزار وجود دارد (مثلاً [LIT89][2] و [ROO90][3]). اگر چه تاکنون رابطهٔ غیرقابل انکاری بین آنها یافت نشده است، ملاحظهٔ بعضی مفاهیم ساده‌ای که در عناصر هر دو سیستم به‌کار می‌روند ارزشمند است.

اگر یک سیستم کامپیوتری را مورد ملاحظه قرار دهیم، اندازه‌گیری سادهٔ اعتبار عبارت خواهد بود از فاصلهٔ زمانی تا بروز یک اشکال[4] (MTBF) در جایی‌که:

$$MTBF=MTTF+MTTR$$

سر واژه‌های MTTF و MTTR فاصلهٔ زمانی تا بروز یک اشکال و فاصلهٔ زمانی برطرف کردن آن می‌باشند.

چرا متوسط زمان بین دو شکست (MTBF) متریک مناسبتری نسبت به تعداد شکست ها بر خطوط برنامه (DEFECT/KLOC) می باشد؟

بسیاری محققان معتقدند که MTBF یک اندازه‌گیری بسیار مفیدتر از KLOC یا FP در مورد نقایص است. آنها این بیان ساده را مطرح می‌کنند که یک مصرف‌کنندهٔ نهایی با خرابی‌ها سر و کار دارد، نه با محاسبه خطای کلی. از آن‌جا که هر خطایی که در یک برنامه موجود است دارای نرخ خرابی یکسانی نمی‌باشد، محاسبهٔ خطای کلی نشان اندکی از اعتبار سیستم بدست می‌دهد. به‌عنوان مثال برنامه‌ای را در نظر بگیرید که به مدت ۱۴ ماه در حال اجرا بوده است. بسیاری از خطاها قبل از این‌که کشف گردند ممکن است به مدت چندین دهه در این برنامه دست نخورده باقی بمانند. MTBF چنان خطاهای مبهمی ممکن است ۵۰ یا حتی ۱۰۰ سال باشد و چون هنوز آشکار نشده‌اند ممکن است نرخ خرابی آنها ۱۸ یا ۲۴ ماه باشد. اگر هریک از خطاهای اولین طبقهٔ خطاها برطرف شده باشند (آنهایی که دارای MTBF) برطرف شده باشند، اثر آنها بر اعتبار نرم‌افزار اندک است.

افزون بر یک میزان اعتبار، ما باید یک میزان قابلیت دسترسی نیز به‌وجود آوریم. قابلیت دسترسی نرم‌افزار[5] احتمال عملکرد یک برنامه بر طبق نیازمندیهای خاص در یک نقطهٔ زمانی است و به‌صورت زیر تعریف می‌شود:

$$\text{Availability} = [MTTE/(MTTF+MTTR)] \times 100\ \%$$ (قابلیت دسترسی)

1.Alvin, W.H.

2.Littlewood, B.

3.Rock, J.

4.Meantime – Between – Failure(MTBF)

5.Software availability

میزان اعتبار **MTBF** نیز به‌همین میزان به **MTTF** و **MTTR** حساس است. میزان قابلیت دسترسی تا اندازهٔ بیشتری به **MTTR** حساس است، که میزان غیرمستقیمی از قابلیت نگهداری نرم‌افزار است.

۸-۸-۲ ایمنی نرم افزار

لوسن **[LEV86]**[1] با نوشتن مطلب زیر تأثیر نرم‌افزار در سیستم‌های حساس ایمنی را مورد بحث قرار می‌دهد:

قبل از این‌که نرم‌افزار در سیستم‌های حساس ایمنی به‌کار برده شود، اغلب به‌وسیلهٔ ابزارهای سنتی (غیرقابل برنامه‌ریزی) مکانیکی و الکترونیک کنترل می‌شد. فنون ایمنی سیستم برای سازگاری با خرابی‌های اتفاقی این سیستم‌های [غیرقابل برنامه‌ریزی] طراحی شده‌اند.

در این‌جا خطاهای طراحی انسانی مورد ملاحظه قرار نگرفته‌اند، زیرا فرض بر این است که همهٔ عیوب مربوط به خطاهای انسانی قبل از عرضه و آغاز به‌کار نرم‌افزار کاملاً قابل پیش‌گیری یا برطرف کردن هستند.

نقل قول

من نمی توانم شرایط را تصور کنم که این کشتی غرق شود. کشتی سازی مدرن بسیار فراتر از این امر است. ای آی اسمیت، کاپیتان تایتانیک

وقتی که نرم‌افزار به‌عنوان بخشی از سیستم کنترل به‌کار گرفته می‌شود، پیچیدگی می‌تواند به اندازهٔ خاصی و یا بیشتری افزایش یابد. کشف عیوب طراحی ظریف ناشی از خطای انسانی ـ چیزی که در کنترل متعارف سخت‌افزاری می‌تواند کشف گردند یا از بین برود ـ وقتی که نرم‌افزار به‌کار گرفته می‌شود بسیار مشکل‌تر می‌گردد.

ایمنی نرم‌افزار[2] یک فعالیت تضمین کیفیت نرم‌افزار است که بر شناسایی و ارزیابی خطرات بالقوه‌ای که ممکن است بر روی نرم‌افزار اثر منفی داشته باشند و موجب خرابی یک سیستم کلی گردند، متمرکز دارد. اگر بتوان خطرات را در اوایل فرآیند مهندسی نرم‌افزار شناسایی کرد، ویژگی‌های طراحی نرم‌افزار را می‌توان به‌گونه‌ای مشخص نمود که به از بین رفتن یا کنترل خطرات بالقوه بپردازد.

یک فرآیند مدل‌سازی و تجزیه و تحلیل به‌عنوان بخشی از ایمنی نرم‌افزار اجرا شده است. نخست، خطرات شناسایی می‌شوند و تحت‌عنوان بحرانی و خطرناک طبقه‌بندی می‌شوند. به‌عنوان مثال، بعضی از خطراتی که به کنترل موتور کامپیوتری یک وسیلهٔ نقلیهٔ موتوری ممکن است مربوط باشد عبارتند از:

- موجب افزایش سرعت غیرقابل کنترلی شود که نمی‌توان آن‌را متوقف کرد.
- به فشار دادن پدال ترمز جواب نمی‌دهد (به‌وسیلهٔ خاموش کردن)؛
- با فعال کردن سویچ درگیر نمی‌شود؛
- کم‌کم از سرعت کاسته می‌شود یا سرعت می‌گیرد.

1.Leveson, N.G.

2.Software safety

بعد از این‌که سطح خطرات این سیستم معین گردید، فنون تجزیه و تحلیل برای تعیین شدت یا احتمال اتفاق افتادن آنها به‌کار گرفته می‌شوند.[1] به‌منظور کارآمد بودن نرم‌افزار باید آن را در بافت کل سیستم مورد تجزیه و تحلیل قرار داد. به‌عنوان مثال، یک خطای ظریف ورودی توسط کاربر (مردم اجزای سیستم هستند) ممکن است به‌وسیلهٔ یک نقص نرم‌افزاری برای تولید اطلاعات کنترلی افزایش یابد، و این در حالی است که این کار با قرار دادن نامناسب یک ابزار مکانیکی انجام گرفته است. اگر مجموعه‌ای از شرایط محیطی بیرونی فراهم آیند، قرار دادن نامناسب آن ابزار مکانیکی می‌تواند موجب خرابی فاجعه‌باری شود. فنون تجزیه و تحلیلی چون تجزیه و تحلیل درختی خرابی[2] [VES81][3]، منطق زمان واقعی[4] [JAN86][5] یا مدل‌های پتری نت[6] [LEV87][7] برای پیش‌بینی زنجیرهٔ اتفاقاتی که می‌توانند خطرآفرین باشند قابل کاربردند؛ این تکنیک‌ها را همچنین برای پیش‌بینی میزان احتمال اتفاق افتادن هر خطر، که زنجیره‌ای از آنها را به وجود می‌آورد، می‌توان به‌کار برد.

بعد از این‌که خطرات شناخته گردیدند و مورد تجزیه و تحلیل قرار گرفتند، نیازمندیهای مربوط به ایمنی مورد نیاز نرم‌افزار قابل تعیین است. به این معنی که مشخصات سیستم می‌تواند در بردارندهٔ لیستی از اتفاقات ناخواسته باشد، و سیستم موردنظر می‌تواند به این اتفاقات پاسخ دهد. به این ترتیب نقش نرم‌افزار در برخورد با اتفاقات ناخواسته شناخته می‌گردد.

اگر چه اعتبار نرم‌افزار و ایمنی نرم‌افزار ارتباط نزدیکی با هم دارند، درک تفاوت ظریف موجود بین آنها دارای اهمیت است. اعتبار نرم‌افزار برای تعیین احتمال این‌که یک خرابی نرم‌افزاری اتفاق بیافتد از تجزیه و تحلیل آماری استفاده می‌کند، با این وجود رخ دادن یک خرابی الزاماً منجر به یک خطر یا اتفاق s سوء نمی‌شود. ایمنی نرم‌افزار به آزمون راه‌هایی می‌پردازد که از طریق آنها خرابی‌ها منجر به شرایطی می‌شوند که ممکن است به یک اتفاق سوء بیانجامند. این بدان معنی است که خرابی‌ها در خلاء در نظر گرفته نشده‌اند، بلکه در بافت یک سیستم کلی کامپیوتری ارزشیابی شده‌اند.

بحث جامع ایمنی نرم‌افزار از حوصلهٔ این کتاب خارج است، خوانندگانی که دارای علاقهٔ بیشتری به این زمینه دارند باید به کتاب لیونسون [LEV95][8] در این زمینه مراجعه کنند.

ارجاع به وب

مقاله ارزشمندی در خصوص سلامت و امن بودن نرم افزار و یک لغت نامه مفصل در آدرس زیر وجود دارد :
www.istcorp.com/hotlist/topics-safety.html

پرسش

چه سوالات اساسی در یک " وارسی پیکربندی " باید پرسیده شوند؟

---

۱. این رهیافت با رهیافت تحلیل ریسک مدیریت پروژه نرم افزاری ، که در فصل ۶ توضیح داده شده ، قابل قیاس است. تفاوت اصلی، تاکید بر فنآوری در مقابل عناوین مرتبط پروژه می باشد.

2.fault tree analysis

3.Vesely,W.E.

4.real-time logic

5.Jahanian, F.

6.petri net models

7.Leveson, N.G.

8.Leveson, N.G.

۸-۹ مصونیت نرم افزار در برابر اشتباه (نرم افزار ضد خطا)

اگر ویلیام شکسپیر توصیه‌ای در مورد وضعیت کنونی به مهندسان نرم‌افزار کرده بود، ممکن بود بنویسد:

"خطا کردن کار انسان است، یافتن سریع خطا و تصحیح آن کاری فرا انسانی است."

در دهۀ ۶۰ قرن بیستم یک مهندس صنعتی ژاپنی، شیگو شینگو [SHI 86][1]، که در کمپانی تویوتا کار می‌کرد، یک تکنیک تضمین کیفیت بوجود آورد که نتیجۀ آن پیش‌گیری یا/ و تصحیح زود، هنگام خطاها در روند ساخت بود. مفهوم مورد کشف شینگو که پوکا ـ یوک (مصونیت از اشتباه) گفته می‌شد، از ابزارهای پوکا ـ یوک[2] استفاده کرد ـ مکانیزم‌هایی که منجر به (۱) جلوگیری از مسایل کیفی بالقوه قبل از رخ دادن آنها شد، یا (۲) تشخیص سریع مسایل کیفی در صورتی‌که وارد کار شده باشند، می‌شد. ما در زندگی هر روزۀ خود با ابزارهای پوکا ـ یوک برخورد داریم (اگر چه ما از این مفهوم بی‌اطلاعیم). بعنوان مثال، سویچ روشن کردن یک وسیلۀ نقلیۀ موتوری اگر یک جعبه دندۀ اتوماتیک در دنده باشد کار نخواهد کرد (یک ابزار جلوگیری[3])؛ بوق اخطار یک ماشین در صورتی‌که کمربند ایمنی بسته نشده باشد به صدا درخواهد آمد (یک ابزار تشخیص[4]).

یک ابزار پوکا ـ یوک کاراً دارای یک‌سری ویژگی های مشترک است:

- **ساده و ارزان است.** اگر ابزاری بسیار پیچیده یا گران باشد، دارای کارآیی نخواهد بود.
- **بخشی از فرآیند کار است.** یعنی ابزار پوکا ـ یوک در درون یک فعالیت مهندسی گنجانده می‌شود.
- **آن را در نزدیکی آن وظیفه از فرآیند، قرار می‌دهد جایی‌که اشتباهات رخ می‌دهند.** و به این وسیله باز خورد و تحصیح خطای سریع ممکن می‌شود.

اگر چه در اصل پوکا یوک برای استفاده در "کنترل کیفی صفر" و برای سخت‌افزارهای ساخته شده بوجود آمد [SHI 86][5]، اما آن را برای استفاده در مهندسی نرم‌افزار نیز می‌توان به‌کار برد. برای روشن ساختن موضوع، ما مسئله زیر را مورد ملاحظه قرار می‌دهیم [ROB97][6]، یک کمپانی تولیدکنندۀ نرم‌افزار، نرم‌افزارهای درخواست شده را به یک بازار جهانی می‌فروشد، فهرست گزینش عمودی و یادمان

1.Shigeo Shingo
2.poka-yoke
3.a prevention device
4.a detection device
5.Shigeo Shingo
6.Robinson, H.

مربوط به هر کاربرد باید به زبان بومی باشند. مثلاً، گزینش انگلیسی برای واژهٔ "Close" یادمان "C" است. وقتی‌که این کاربرد در یک کشور فرانسه زبان فروخته می‌شود همین گزینش واژهٔ "Fermer" با یادمان " F" است. برای تهیه کردن یک گزینش مناسب برای هر منطقهٔ بومی یک "بومی‌گو" (فردی مطلع که از زبان بومی و اصطلاحات آن آگاه است) صورت گزینش واژه‌ها را به‌طور مناسب ترجمه می‌کند. مشکل این است که باید مطمئن شویم (۱) مدخل گزینشی (می‌تواند صدها مدخل وجود داشته باشد) با استانداردهای مناسب هم‌خوانی دارد و صرف‌نظر از زبانی که به‌کار برده می‌شود، تضادی وجود ندارد.

کاربرد پوکا ـ یوک برای آزمودن صورت‌های کاربردی گوناگونی که در زبان‌های مختلف اجرا شده‌اند، همان‌گونه که در بالا توصیف شد، در مقاله‌ای توسط هری رابینسون[ROB 97] مورد بحث واقع شده است:[1]

نخست ما تصمیم گرفتیم تا مشکل آزمودن صورت گزینشی (منو) را به بخش‌هایی که برای ما قابل حل باشند تقسیم کنیم. اولین پیشرفت ما در این مسئله درک این نکته بود که دو جنبهٔ مجزا در رابطه با کاتالوگ پیام وجود دارد. یکی جنبهٔ تمای آن بود: ترجمه‌های سادهٔ متن، مثل تغییر "Close" به "Fermer"، به‌دلیل این‌که تیم آزمون‌کننده نمی‌توانست تمام ۱۱ زبان مورد نظر را به روانی صحبت کند، ما باید این جنبه را به متخصصان زبان ارجاع می‌دادیم.

دومین جنبهٔ کاتالوگ‌های پیام، ساختار خاص آن بود: قواعد نحوی‌ای که یک کاتالوگ دارای ساختار مناسب، باید از آنها تبعیت کند. برخلاف محتوا، بررسی کردن جنبه‌های ساختاری کاتالوگ‌ها برای تیم آزمون‌کننده مقدور می‌باشد.

به‌عنوان یک مثال برای درک معنی ساختار برچسب‌ها و یادمان‌های یک صورت گزینش (منو) را ملاحظه کنید. یک صورت گزینش واژه‌ای از برچسب‌ها و یادمان‌های مربوط به آن تشکیل یافته است. هر صورت گزینش واژه‌ای، صرف‌نظر از محتوا یا منطقهٔ زبانی آن، باید از قواعد زیر که در راهنمای سبک محتوا برشمرده شده‌اند، پیروی کند:

- *هر یادمان باید در برچسب مربوط به آن گنجانده شود.*
- *هر یادمان در صورت گزینش واژه‌ای باید منحصربه‌فرد باشد.*
- *هر یادمان باید یک کاراکتر تنها باشد.*
- *هر یادمان باید در ASCII وجود داشته باشد.*

این قواعد برای تمام مناطق زبانی یکسان هستند، و برای آزمون این‌که یک صورت گزینش واژه‌ای به‌صورت مناسب نسبت به زبان منطقه‌ای موردنظر ساختاربندی شده است، به‌کار می‌رود.

امکانات گوناگونی برای چگونگی مصون کردن یادمان‌های صورت گزینش واژه‌ای از اشتباه وجود داشت.

ارجاع به وب

یک مجموعه کامل از منابع پوکا-یوک در آدرس زیر قابل دسترسی است:
www.campbell.berry.edu/faculty/igrout/pokayoke.shtml

ابزار پیش‌گیری. ما می‌توانستیم برنامه‌ای برای تولید خودکار یادمان‌ها بنویسیم، و برای این کار از برچسب‌های موجود در هر صورت گزینش واژه‌ای استفاده کنیم. این رهیافت می‌توانست از اشتباهات پیش‌گیری کند، اما مسئلهٔ انتخاب یک یادمان خوب مشکل است و تلاشی که برای نوشتن این برنامه مورد نیاز است با سودی که عاید خواهد شد قابل توجیه نخواهد بود.

ابزار پیش‌گیری. ما می‌توانستیم برنامه‌ای بنویسیم که می‌توانست از این مسئله که بومی‌گو یادمانی را انتخاب کند که با معیارهای موردنظر سازگاری ندارند جلوگیری کند. این رهیافت نیز می‌توانست از اشتباهات پیش‌گیری کند، اما سود حاصله حداقل خواهد بود؛ تشخیص و اصلاح یادمان‌های ناصحیح بعد از اتفاق افتادن آسان است.

ابزار تشخیص. ما می توانستیم برنامه‌ای فراهم کنیم تا بتوانیم بررسی کنیم که آیا یادمان‌ها و برچسب‌های صورت گزینش واژه‌ای با معیارهای موردنظر سازگارند یا نه. بومی‌گویان ما می‌توانستند در ترجمه‌های خود قبل از فرستادن کاتالوگ‌ها به ما، این برنامه‌ها را اجرا کنند. این رهیافت امکان بازخورد سریع را فراهم می‌آورد و احتمال وجود آن به‌عنوان یک قدم بعدی متحمل است.

ابزار تشخیص. ما می‌توانستیم برنامه‌ای برای آزمودن برچسب‌ها و یادمان‌های صورت گزینش واژه‌ای بنویسیم، و بعد از این‌که کاتالوگ‌ها به‌وسیلهٔ بومی‌گویان به ما برگشت داده شد این برنامه را در مورد آنها اجرا کنیم. این رهیافت مسیری است که اکنون در پیش گرفته‌ایم. این رهیافت به کارآیی بعضی از روش‌هایی که در بالا توضیح داده شد نیست، در این نقطه نیز به آسانی قابل تصحیح است.

چندین متن پوکا ـ یوک کوچک به‌عنوان ابزارهای پوکا ـ یوک به‌کار برده شدند تا جنبه‌های ساختاری صورت گزینش واژه‌ای تثبیت گردد. یک متن پوکا ـ یوک کوچک می‌تواند جدول را بخواند، یادمان‌ها و برچسب‌ها را از کاتالوگ بازیابی کند، و زنجیره‌های بازیافته را با معیارهای موردنظر ذکر شده بالا مقایسه نماید.

متن‌های پوکا ـ یوک کوتاه بودند (تقریباً ۱۰۰خط)، نوشتن آنها آسان بود (بعضی از آنها در کمتر از یک ساعت نوشته شده‌اند) و اجرای آنها نیز ساده بود. ما پوکا ـ یوک خود را در مقایسه با ۱۶ کاربرد موجود در متن بدون نقص انگلیسی و نیز ۱۱ متن خارجی اجرا کردیم. هر منطقه دارای ۱۰۰ صورت گزینش واژه‌ای بود و در کل ۱۲۰۰ صورت گزینش واژه‌ای وجود داشت. ابزارهای پوکا ـ یوک ۳۱۱ اشتباه در صورت‌های گزینش واژه‌ای و یادمان‌ها یافتند. تعداد اندکی از مسایلی که ما آشکار ساختیم مسایل ریشه‌ای بودند، اما در کل آنها می‌دانند موجب دردسرهای زیادی در آزمودن و اجرای درخواست‌های محلی شدهٔ ما باشند.

---

۷. پاراگرافها، با کسب اجازه از نویسنده اثر [ROB97] و با اندکی تصرف آورده شده است.

مثالی که در بالا توصیف گردید یک ابزار پوکا ـ یوک را توصیف می‌کند که وارد فعالیت آزمون مهندسی نرم‌افزار شده است. تکنیک پوکا ـ یوک را می‌توان در سطوح طراحی، برنامه‌نویسی و آزمودن به‌کار برد و این تکنیک می‌تواند یک فیلتر تضمین کیفیت کاراً بدست دهد.

۸-۱۰ استانداردهای کیفیت ایزو ۹۰۰۱ [1]

ارجاع به وب

اتصالات توسعه یافته به منابع ایزو ۹۰۰۱/۹۰۰۰ در آدرس زیر وجود دارد: www.tartara.ab.ca /iso-list.html

این بخش چندین هدف دارد، هدف اصلی آن توصیف استاندارد بین‌المللی، مهم و در حال توسعهٔ **ISO 9001** است. این استاندارد، که توسط بیش از ۱۳۰ کشور پذیرفته شده است، به‌طور مستمر بر اهمیت آن به‌عنوان ابزاری که به‌وسیلهٔ آن مشتریان می‌توانند توانایی سازندگان نرم‌افزار را مورد قضاوت قرار دهند افزوده شده است. یکی از مشکلات مربوط به سری استانداردهای **ISO 9001** مربوط نبودن آنها به صنعت خاصی است: این استاندارد با اصطلاحات عمومی بیان گردیده است، و به‌وسیلهٔ تولیدکنندگان محصولات مختلفی چون بلبرینگ، مو خشک‌کن، اتومبیل، وسایل ورزشی، تلویزیون، و نیز نرم‌افزار قابل تعبیر است. مدارک چندی به‌وجود آمده است که این استاندارد را به صنعت نرم‌افزار مربوط می‌کند، اما به بسیاری از جزییات پرداخته شده است. هدف این بخش توصیف معنی **ISO 9001** در رابطه با عناصر کیفیت و فنون توسعه است.

۸-۱۰-۱ رهیافت ایزو جهت سیستم های تضمین کیفیت

برای صنعت نرم‌افزار استانداردهای مربوطه عبارتند از:

ایزو ۹۰۰۰ تشریح می کند که چه کاری باید انجام شود تا مطلوب باشد، اما چگونگی آن را توضیح نمی دهد.

- سیستم‌های کیفیت **ISO 9001** ـ مدلی برای تضمین کیفیت در طراحی، توسعه، تولید، نصب و خدمات‌دهی است. این استانداردی است که سیستم کیفیت به‌کار برده شده برای تضمین توسعهٔ محصول مورد طراحی را توصیف می‌کند.
- **ISO ۹۰۰۰** رهنمودهایی است برای به‌کارگیری **ISO ۹۰۰۱** در کار توسعه و در عرضه و نگهداری نرم‌افزار و سند خاصی است که **ISO ۹۰۰۱** را برای تولیدکنندهٔ نرم‌افزار معنی می‌کند.
- **ISO ۹۰۰۴** مدیریت کیفیت و عناصر سیستم کیفیت ـ قسمت دوم این سند رهنمودهایی برای خدمات تسهیلات نرم‌افزاری، چون حمایت از مصرف‌کننده فراهم می‌کند.

این نیازمندی‌ها در زیر ۲۰ عنوان گروه‌بندی شده‌اند:

مسئولیت مدیریت — تجهیزات آزمون و اندازه‌گیری و بازرسی

---

۱. این بخش توسط مایکل استوسکی تحریر گردیده است، که مطابق اصول ایزو ۹۰۰۰ و استاندارد ایزو ۹۰۰۱ شکل یافته است. کتابهای کاربردی برای مهندسی نرم افزار و یک فیلم ویدئویی که توسط راجر اس. پرسمن و تشکیلاتش، (با کسب اجازه)تهیه شده است.

سیستم کیفیت	بازرسی و وضعیت آزمون
بررسی قرارداد	اقدام اصلاحی
کنترل طرح	کنترل بر محصول ناسازگار
کنترل سند	جابه‌جایی، انبار، بسته‌بندی و عرضه
خرید	ثبت کیفیت
محصول عرضه شدهٔ خریدار	بازرسی درونی کیفیت
قابلیت پی‌گیری و تشخیص تولید	آموزش
کنترل مرحله‌ای	سرویس کردن
بازرسی و آزمون	فنون آماری

جا دارد که نگاهی به گلچینی از کاربردهای ISO ۹۰۰۱ داشته باشیم. این قسمت چگونگی عملکرد ISO ۹۰۰۱ را در مورد QA و روند توسعه برای خواننده مشخص می‌کند. این گزیده از بخش ۱۱-۴ انتخاب شده و در قسمت بعدی می‌آید.

۸-۱۰-۲ استاندارد ایزو ۹۰۰۱

به‌منظور حفظ مطابقت محصول با نیازمندیهای ساخت آن، تولیدکننده باید به کنترل، تعیین و حفظ بازرسی، اندازه‌گیری و فراهم کردن و سایل آزمون محصول بپردازد، چه آنها را در تملک خود داشته باشد، چه به‌صورت امانت و چه آنها را خریدار، فراهم کرده باشد. این تجهیزات باید به‌گونه‌ای به‌کار برده شوند که نشان دهند موارد مبهم اندازه‌گیری، شناخته گردیده‌اند و با توانایی‌های اندازه‌گیری مورد نیاز سازگاری دارند.

ایزو ۹۰۰۰ برای نرم افزار

اولین چیزی که باید بدان توجه کرد عمومیت آن است؛ آن را می‌توان در مورد تولیدکنندهٔ هر محصولی به‌کار برد. دومین چیزی که باید بدان توجه کرد، دشواری تعبیر پاراگراف است ـ این پاراگراف به روشنی بر مراحل مهندسی استاندارد تمرکز دارد، و در حالی‌که تجهیزاتی چون ترموکوپل‌ها، کالیبرسنج‌ها و پتانسیل‌مترها رایج‌تر هستند. یکی از تعابیر این پاراگراف این است که عرضه‌کننده باید مطمئن شود که هر وسیلهٔ نرم‌افزاری که برای آزمودن به‌کار می‌رود، دست کم دارای کیفیت نرم‌افزار موردنظر است، و نیز او باید مطمئن شود که وسایل آزمونی که ارزش‌های اندازه‌گیری را به‌وجود می‌آورد، به‌عنوان مثال مونیتور عملکرد، در مقایسه با دقت موردنظر برای عملکرد، در تصریح نیازمندیهای، دارای دقت قابل قبولی هستند.

طرح تضمین کیفیت نرم افزار (SQA)

۸-۱۱ طرح تضمین کیفیت نرم افزار (SQA)

طرح SQA[1] راهنمایی عملی برای به‌وجود آوردن تضمین کیفیت نرم‌افزاری بدست می‌دهد. این طرح که به‌وسیلهٔ گروه SQA به‌وجود آمده است، به‌عنوان الگویی برای فعالیت‌های SQA عمل می‌کند که برای هر پروژهٔ نرم‌افزاری برقرار گردیده‌اند.

استانداردی به‌وسیلهٔ IEEE برای طرح‌های SQA پیشنهاد شده است [IEE 94]. بخش‌های اولیه، هدف و حوزهٔ عمل این مدرک را توصیف کردند، و آن فعالیت‌های مرحله‌ای را که به‌وسیلهٔ تضمین کیفیت پوشش داده شده‌اند را نشان دادند. تمام مدارکی که در طرح SQA به آنها اشاره شده است برشمرده شده‌اند و تمام استانداردهای قابل کاربرد ذکر گردیده‌اند. بخش مدیریت طرح جایگاه SQA را در ساختار سازمانی، کارهای SQA و فعالیت‌ها و جایگاهشان در فرآیند نرم‌افزاری، و نقش‌های سازمانی و مسئولیت‌های مربوط به کیفیت محصول را توصیف می‌کند.

بخش مستندسازی (یا رجوع) به هر یک از محصولات کاری که به‌عنوان بخشی از فرآیند نرم‌افزاری تولید شده‌اند به توصیف آنها می‌پردازد. این موارد عبارتند از:

- اسناد پروژه (مثلاً، طرح پروژه)
- مدل‌ها (مثلاً، ERDها، سلسله مراتب طبقه‌ای)
- اسناد فنی (مثلاً، مشخصات، طرح‌های آزمون)
- اسناد کاربر (مثلاً، فایل‌های کمک و راهنمایی)

افزون بر این، این بخش مجموعه حداقل محصولات کاری مورد قبول برای رسیدن به کیفیت بالا را تعریف می‌کند.

لیست استانداردها، تمرینات، کنوانسیون‌ها و همهٔ استانداردهای/ تمرینات قابل کاربردی که در مدت فرآیند نرم‌افزاری به‌کار برده شده‌اند (مثل، استانداردهای سند، استانداردهای برنامه‌نویسی، و رهنمودهای بررسی) و نیز تمام متریک‌های پروژه‌ها و مراحل و (در بعضی موارد) محصولات که به‌عنوان بخشی از کار مهندسی نرم‌افزاری باید گردآوری شوند، برشمرده شده‌اند.

بخش بررسی‌ها و وارسی‌های طرح، بررسی‌ها و وارسی‌هایی را که توسط تیم مهندسی نرم‌افزاری، گروه SQA، و خریدار باید به‌عمل آیند را مشخص می‌کند. این بخش مرور کلی رهیافت طرح را برای هر بررسی و وارسی بدست می‌دهد.

بخش آزمون به روال و طرح آزمون نرم‌افزار (فصل ۱۸) نظر دارد. این بخش همچنین نیازمندیهای نگهداری نتایج ثبت شدهٔ آزمونهای را مشخص می‌کند. گزارش مشکل و اقدام اصلاحی، مراحل گزارش،

1.Schoonmaker, S.J.

پی‌گیری و برطرف کردن اشکالات و نقایص را مشخص می‌کند، و مسئولیت‌های سازمانی در برابر این فعالیت‌ها را مشخص می‌کند.

قسمت‌های دیگر طرح SQA ابزارها و روش‌های تقویت فعالیت‌ها و کارهای SQA را مشخص می‌کنند، مراحل مدیریت طرح نرم‌افزار برای کنترل کردن تغییر را مشخص می‌کنند. یک رهیافت مدیریت قرارداد را تعریف می‌کنند. روش‌هایی برای گردآوری، محافظت و حفظ مدارک مشخص می‌کنند. آموزش‌های لازم برای برآورده کردن اهداف طرح را مشخص می‌کنند، و روش‌هایی را برای تشخیص، ارزیابی، نظارت و کنترل خطرات مشخص می‌کنند.

۸-۱۲ خلاصه

تضمین کیفیت نرم‌افزاری یک "فعالیت چتری" است که در هر یک از مراحل فرآیند نرم‌افزاری به‌کار بسته شده است. SQA دربردارندهٔ مراحل به‌کارگیری مؤثر روش‌ها، بررسی‌های فنی منظم، تکنیک‌ها و راهبردهای آزمودن ؛ ابزارهای یوکا- یوک؛ روال‌هایی برای کنترل تضمین مطابقت با استانداردها، و سنجش و مکانیزم‌های گزارش کردن می‌باشد.

طبیعت پیچیدهٔ کیفیت نرم‌افزاری موجب پیچیدگی SQA شده است - یکی از خواص برنامه‌های کامپیوتری که به‌عنوان "مطابقت با نیازمندیهای تعریف شدهٔ صریح و ضمنی" تعریف شده است. اما وقتی کیفیت نرم‌افزاری را به‌صورت عمومی تعریف کنیم بسیاری از محصولات مختلف، فاکتورهای پردازشی و متریک‌های مربوطه را در برمی‌گیرد.

بررسی‌های نرم‌افزاری یکی از مهم‌ترین فعالیت‌های SQA می‌باشند. بررسی‌ها به‌عنوان فیلترهایی در تمام فعالیت‌های مهندسی نرم‌افزاری عمل می‌کنند، که باعث برطرف شدن خطاها، زمانی‌که آنها برای یافتن و برطرف کردن گسترش زیادی نیافته‌اند، خواهند شد. بررسی فنی منظم یک گردهم‌آیی روش‌مند است که معلوم شده، برای مشخص کردن خطاها بسیار مؤثر است.

برای تضمین مناسب کیفیت نرم‌افزار، اطلاعاتی دربارهٔ فرآیند مهندسی نرم‌افزار باید گردآوری، ارزشیابی و پخش گردد. SQA آماری به بهتر کردن کیفیت محصول و خود فرآیند نرم‌افزار کمک می‌کند. مدل‌های اعتبار نرم‌افزار اندازه‌گیری‌هایی را بسط می‌دهند که برآورد و ارزیابی داده‌های گردآوری شده در مورد نقایص را در قالب نرخ‌های ناکامی موجود در طرح و پیش‌بینی‌های اعتبار، ممکن می‌سازد.

به‌طور خلاصه، ما سخنان دان و آلمن را بازگو می‌کنیم. [DUN82][1]: "تضمین کیفیت نرم‌افزاری ترسیم مقررات مدیریتی و اصول طراحی تضمین کیفیت در فضای قابل کاربرد فنی و مدیریتی مهندسی نرم‌افزار است." رسیدن اطمینان از کیفیت یکی از معیارهای دیسیپلین و انضباط مهندسی کامل است. اگر آن‌چه در بالا بدان اشاره گردید با موفقیت انجام شود، مهندسی نرم‌افزار کامل به‌دنبال خواهد بود.

1.Schmauch, C.H.

مسایل و نکاتی برای تفکر و تعمق بیشتر

۸-۱ در ابتدای این فصل متذکر شدیم که «کنترل تغییرات، قلب کنترل کیفیت است». با نظر به این نکته که ایجاد هر برنامه‌ای با برنامه دیگر متفاوت است، باید در جستجوی چه تغییراتی باشیم و آنها را چگونه کنترل کنیم؟

۸-۲ اگر مشتری، تغییراتی در مقاصد خویش داشته باشد، آیا ارزیابی کیفیت نرم‌افزار ممکن خواهد بود؟

۸-۳ کیفیت و قابلیت اطمینان مفاهیمی مرتبط به هم می باشند، اما اساساً تفاوتهایی چند دارند. آنها را مورد بحث قرار دهید.

۸-۴ آیا امکان دارد برنامه‌ای صحیح عمل کند، اما هنوز قابل اطمینان نباشد؟ توضیح دهید.

۸-۵ آیا امکان دارد برنامه‌ای صحیح عمل کند، اما کیفیت خوبی را به نمایش نگذارد؟ توضیح دهید.

۸-۶ چرا اغلب بین یک گروه مهندسی نرم‌افزار و یک گروه مستقل تضمین کیفیت نرم‌افزار شکرآب است؟ این رفتار سالمی است؟

۸-۷ فرض کنید مسئولیت بهبود کیفیت نرم‌افزار در سازمان شما، به شما محول گردیده است: اولین چیزی که باید انجام دهید چیست؟ گام بعدی چیست؟

۸-۸ علاوه بر شمارش خطاها، آیا خصوصیات شمردنی دیگری در خصوص کیفیت وجود دارد؟ آنها چه هستند و آیا مستقیماً قابل اندازه گیری می باشند؟

۸-۹ یک مرور رسمی فنی تنها وقتی مؤثر است که همه کس از قبل به بهترین صورت ممکن آماده شده باشند. شما چگونه می‌توانید یک همکار ناآماده را شناسایی کنید؟ اگر شما رهبریت بازبینی و مرور را عهده دار باشید چه خواهید کرد؟

۸-۱۰ بعضی افراد بر این باورند که یک **FTR** باید سبک برنامه سازی را نیز به خوبی درستی آن ارزیابی نماید؟ آیا این ایده خوبی است؟ چرا؟

۸-۱۱ جدول۸-۱را مرور کرده، چهار علت حیاتی خطاهای اساسی و میانه را مشخص کنید. با به‌کار بردن اطلاعات موجود در فصل‌های دیگر اقدامات اصلاحی را پیشنهاد نمایید.

۸-۱۲ یک سازمان فرآیند پنج مرحله ای مهندسی نرم‌افزار را به‌کار می‌برد که در آن‌خطاها بر طبق توزیع درصدی ذیل یافت می‌شوند:

مرحله	درصد خطاهای یافت شده
۱	٪۲۰
۲	٪۱۵
۳	٪۱۵

۴	٪۴۰
۵	٪۱۰

با به‌کار بردن اطلاعات جدول ۸-۱ و این توزیع درصد، شاخص نقص کلی سازمان را محاسبه کنید. فرض کنید PS=100,000.

۸-۱۳ متون قابلیت اعتبار نرم افزار را تحقیق کرده، مقاله‌ای بنویسید که یک مدل قابلیت اطمینان نرم‌افزار را توضیح دهد. حتماً یک مثال ارائه دهید.

۸-۱۴ راه انتقاد از مفهوم **MTBF** برای نرم‌افزار باز است. آقای می‌توانید به دلایل این امر فکر کنید؟

۸-۱۵ دو سیستم حیاتی ایمنی را که به‌وسیلهٔ کامپیوتر کنترل می‌شوند مورد ملاحظه قرار دهید. برای هر کدام حداقل دست‌کم سه خطر نام ببرید که می‌توان آنها را مستقیما" به شکست های نرم‌افزار ربط داد.

۸-۱۶ با به‌کار بردن منابع وب و منابع مکتوب، یک برنامهٔ آموزشی ۲۰ دقیقه‌ای بر پوکا ـ یوک تهیه نموده، در کلاس خود ارائه کنید.

۸-۱۷ چند شیوه پوکا ـ یوک پیشنهاد کنید که ممکن است برای تشخیص یا پیشگیری از خطاهایی که معمولاً قبل از "ارسال" یک پیام پست الکترونیکی با آن مواجه می‌شوید به‌کار آیند.

۸-۱۸ یک نسخه از ایزو ۹۰۰۱ و ایزو ۹۰۰۳ تهیه کنید. یک ارائه داشته باشید که سه نیازمندی ایزو ۹۰۰۱ را بحث کند و چگونگی کاربرد آنها در بافت نرم‌افزاری بحث شود.

فهرست منابع و مراجع

[ALV64] Alvin, W.H. von (ed.), *Reliability Engineering*, Prentice-Hall, 1964.

[ANS87] ANSI! ASQC A3- I 987, Quality Systems Terminology, 1987.

[ART92] Arthur, L.J., *Improving Software Quality: An Insider's Guide to TQM*, Wiley, 1992.

[ART97] Arthur, L.J., "Quantum Improvements in Software System Quality, *CACM*, vol. 40, no. 6, June 1997, pp. 47-52.

[BOE8 I] Boehm, B., *Software Engineering Economics*, Prentice-Hall, 198 I.

[CR079] Crosby, P., *Quality Is Free*, McGraw-Hill, 1979.

[DEM86] Deming, W.E., *Out of the Crisis*, MIT Press, 1986.

[DEM99] DeMarco, T., "Management Can Make Quality (Im)possible," *Cutter IT Summit*, Boston, April 1999.

[DIJ76] Dijkstra, E., *A Discipline of Programming*, Prentice-Hall, 1976.

[DUN82] Dunn, R. and R. Ullman, *Quality Assurance for Computer Software*, McGraw-Hill, 1982.

[FRE90] Freedman, D.P. and G.M. Weinberg, *Handbook of Walkthroughs, Inspections and Technical Reviews*, 3rd ed., Dorset House, 1990.

[GIL93] Gilb, T. and D. Graham, *Software Inspections*, Addison-Wesley, 1993.

[GLA98] Glass, R., "Defining Quality Intuitively,'" *IEEE Software*, May 1998, pp. 103-104, 107.

[HOY98] Hoyle, D., *ISO 9000 Quality Systems Development Handbook: A Systems Engineering Approach*, Butterworth-Heinemann, 1998.

[IBM81] "Implementing Software Inspections," course notes, IBM Systems Sciences Institute, IBM Corporation, 1981.

IEE94] *Software Engineering Standards*, 1994 ed., IEEE Computer Society, 1994.

[JAN86] Jahanian, F. and A.K. Mok, "Safety Analysis of Timing Properties of Real-Time Systems", *IEEETrans. Software Engineering*, vol. SE-12, no. 9, September 1986, pp. 890-904.

[JON86] Jones, T.C., *Programming Productivity*, McGraw-Hill, 1986.

[KAN95] Kan, S.H., *Metrics and Models in Software Quality Engineering*, Addison-Wesley, 1995.

[KAP95] Kaplan, C., R. Clark, and V. Tang, *Secrets of Software Quality: 40 Innovations from IBM*, McGraw-Hill, 1995.

[LEV86] Leveson, N.G., "Software Safety: Why, What, and How," *ACM Computing Surveys*, vol. 18, no. 2, June 1986, pp. 125-163.

[LEV87] Leveson, N.G. and J.L. Stolzy, "Safety Analysis Using Petri Nets, *IEEE Trans. Software Engineering*, vol. SE-13, no. 3, March 1987, pp. 386-397.

[LEV95] Leveson, N.G., *Safeware: System Safety And Computers*, Addison-Wesley, 1995.

[LIN79] Linger, R., H. Mills, and B. Witt, *Structured Programming*, Addison-Wesley, 1979.

[LIT89] Littlewood, B., "Forecasting Software Reliability," in *Software Reliability: Modelingand Identification* (S. Bittanti, ed.), Springer-Verlag, 1989, pp. 141-209.

[MUS87] Musa, J.D., A. Iannino, and K. Okumoto, *Engineering and Managing Software with Reliability Measures*, McGraw-Hill, 1987.

[POR95] Porter, A., H. Siy, C.A. Toman, and loG. Votta " An Experiment to Assess the Cost-Benefits of Code Inspections in Large Scale Software Development," Proc. *Third ACM SIGSOFT Symposium on the Foundations of Software Engineering*, Washington, D.C., October 1996, ACM Press, pp. 92-103.

[ROB97] Robinson, H., "Using poka- Yoke Techniques for Early Error Detection,"' *Proc. Sixth International Conference on Software Testing Analysis and Review* (STAR'97) , 1997, pp. 119-142.

[RO090] Rook, *J., Software Reliability Handbook*, Elsevier, 1990.

[SCH98] Schulmeyer, G.C. and].1. McManus (eds.). *Handbook of Software Quality Assurance*, 3rd ed., Prentice-Hall, 1998.

[SCH94] Schmauch, C.H., *ISO 9000 for Software Developers*, ASQC Quality Press, 1994.

[SCH97] Schoonmaker, S.]., *ISO 9001 for Engineers and Designers*, McGraw-Hill, 1997.

[SHI86] Shigeo Shingo, *Zero Quality Control: Source Inspection and the Poka-yoke System*, Productivity Press, 1986.

[SOM96] Somerville, I., *Software Engineering*, 5th ed., Addison-Wesley, 1996.

[VES81] Veseley, W.E., et al., *Fault Tree Handbook*, U.S. Nuclear Regulatory Commission, NUREG-0492, January 1981.

خواندنیهای دیگر و منابع اطلاعاتی

Books by Moriguchi *(Software Excellence: A Total Quality Management Guide*, Productivity Press, 1997) and Horch *(Practical Guide to Software Quality Management*, Artech Publishing, 1996) are excellent management-level presentations on the benefits of formal quality assurance programs for computer software. Books by Deming [DEM86] and Crosby [CR079] do not focus on software, but both books are must reading for senior managers with software development responsibility. Gluckman and Roome *(Everyday Heroes of the Quality Movement*, Dorset House, 1993) humanizes quality issues by telling the story of the players in the quality process. Kan *(Metrics and Models in Software Quality Engineering*, Addison-Wesley, 1995) presents a quantitative view of software quality.

Tingley *(Comparing ISO 9000, Malcolm Baldrige, and the SEI CMM for Software*, Prentice-Hall, 1996) provides useful guidance for organizations that are striving to improve their quality management processes. Oskarsson (An *ISO 9000 Approach to Building Quality Software*, Prentice-Hall, 1995) discusses the ISO standard as it applies to software.

Dozens of books have been written about software quality issues in recent years. The following is a small sampling of useful sources:

Clapp, J.A., et al., *Software Quality Control, Error Analysis and Testing*, Noyes Data Corp., 1995.

Dunn, R.H. and RS. Ullman, *TQMforComputerSoftware*, McGraw-Hili, 1994.

Fenton, N., R. Whitty, and Y. Iizuka, *Software Quality Assurance and Measurement: Worldwide Industrial Applications*, Chapman & Hall, 1994.

Ferdinand, A.E., *Systems, Software, and Quality Engineering*, Van Nostrand-Reinhold, 1993.

Ginac, F.P., *Customer Oriented Software Quality Assurance*, Prentice-Hall, 1998.

Ince, D. *ISO 9001 and Software Quality Assurance*, McGraw-Hili, 1994.

Ince, D., *An Introduction to Software Quality Assurance and its Implementation*, McGraw-Hili.

1994.

Jarvis, A. and V. Crandall, *Inroads to software Quality: "How to" Guide and Toolkit*, Prentice-Hall, 1997.

Sanders, J., *Software Quality: A Frameworkfor Success in Software Development*, Addison-Wesley, 1994.

Sumner, F.H., *Software Quality Assurance*, Macmillan, 1993.

Wallmuller, E., *Software Quality Assurance: A Practical Approach*, Prentice-Hall, 1995.

Weinberg, G.M., *Quality Software Management*, four volumes, Dorset House, 1992, 1993, 1994, 1996.

Wilson, RC., *Software Rx: Secrets of Engineering Quality Software*, Prentice-Hall, 1997.

An anthology edited by Wheeler, Brykczynski, and Meeson *(Software Inspection: Industry Best Practice*, IEEE Computer Society Press, 1996) presents useful information on this important SQA activity. Friedman and Voas *(Software Assessment*, Wiley, 1995) discuss both theoretical underpinnings and practical methods for ensuring the reliability and safety of computer programs.

Musa *(Software Reliability Engineering: More Reliable Software, Faster Development and Testing*, McGraw-Hill, 1998) has written a practical guide to applied software reliability techniques. Anthologies of important papers on software reliability have been edited by Kapur et al. *(Contributions to Hardware and Software Reliability Modelling*, World Scientific Publishing Co., 1999), Gritzalis *(Reliability, Quality and Safety of Software-Intensive Systems*, Kluwer Academic Publishers, 1997), and Lyu *(Handbook of Software Reliability Engineering*, McGraw-Hill, 1996). Storey *(safety-Critical Computer Systems*, Addison-Wesley, 1996) and Leveson [LEV95] continue to be the most comprehensive discussions of software safety published to date.

In addition to [SHI86]. the *poka-yoke* technique for mistake-proofing software is discussed by Shingo *(The Shingo Production Management System: Improving Process Functions*, Productivity Press, 1992) and Shimbun *(Poka- Yoke: Improving Product Quality by Preventing Defects*, Productivity Press, 1989).

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

http://www.mhhe.com/engcs/compsci/pressman/resources/sqa.mhtml