مهندسی کیفیت
در فرآیند تحقیق و توسعه‌ی نرم‌افزار

فهرست

1. پیش‌گفتار

بحث کیفیت، و ساز و کارهای آن در «چرخه‌ی حیات توسعه‌ی نرم افزار»[1]، بحث تازه ای نیست و به نوع و یا پیچیدگی محصول یا سرویس خاصی نیز محدود نمی‌شود. مدیران و دست اندرکاران کلیدی درهمه‌ی بخش‌های تولیدی و خدماتی، اعم از سخت افزار، نرم افزار، و دیگرافزارها، دیر یا زود با چالش کیفیت روبرو می‌شوند و فرصت می‌یابند در کنار ظرفیت و توان «کمّی» تولید، میزان تعهد و باور خود به جنبه‌های گوناگون «کیفی» را نیزبه نمایش بگذارند. گروهی کیفیت را دغدغه‌ی اصلی خود نمی‌پندارند و آنرا به بعد موکول می‌کنند؛ عده‌ای گمان می‌کنند کیفیت اصولاً نیاز به تدبیر و برنامه ریزی ندارد و خود به خود و به گونه‌ای تصادفی ظهورمی‌کند؛ گروهی هم هستند که کیفیت را از اولویت‌های نخست (و گاه نخستین اولویت) فرآیند تولید به حساب آورده و لازم می‌بینند اندیشه و سرمایه‌ی جدی خرج آن کنند. اما، حتا در این حالت آخرهم شاهدیم که طرح‌های تولیدی و خدماتی به ندرت به «کیفیت مطلوب» دست می یابند، و این کاستی، به ویژه در حوزه‌ی نرم‌افزار، فراوان دیده می‌شود.

کیفیت مطلوب در نرم‌افزار اساساً چیست و چگونه به دست می‌آید؟ آیا می‌توان به هرسطح از پیش تعیین شده‌ای از کیفیت دست یافت و سپس استمرار و یا ارتقاء آن را در چرخه‌های بعدی تولید تضمین کرد؟ چرا با وجود ابزار پیشرفته‌ی تولید، تکنیک‌های کارآمد تحقیق و توسعه، و فرآیندهای مبسوط و مفصل، تولید نرم‌افزار بی‌عیب و نقص هنوز هم میسر نیست؟

درادامه‌ی این مقاله می‌کوشیم تا حد ممکن به این پرسش‌ها پاسخ گفته و نشان دهیم که رسیدن به کیفیت مطلوب در نرم افزار، و استمرار آن، بدون وجود زیرساختی منسجم مبتنی بر نوعی تفکر کیفی که همه‌ی سطوح سازمان را در برمی‌گیرد، امکان پذیر نیست.

2. «کیفیت مطلوب» در نرم‌افزار

در کنار برداشت‌های آکادمیک، ذهنی، و مبهمی که از عبارت «کیفیت نرم‌افزار» درمیان نرم‌افزارسازان رایج هستند و غالباً نیز به مشاجره‌های کسل کننده و بی‌نتیجه کشیده می‌شوند، سازمان‌های استانداردسازی، انستیتوهای پژوهشی، و انجمن‌های علمی و حرفه‌ای نظیر IEEE, ISO, ITIL, SEI,

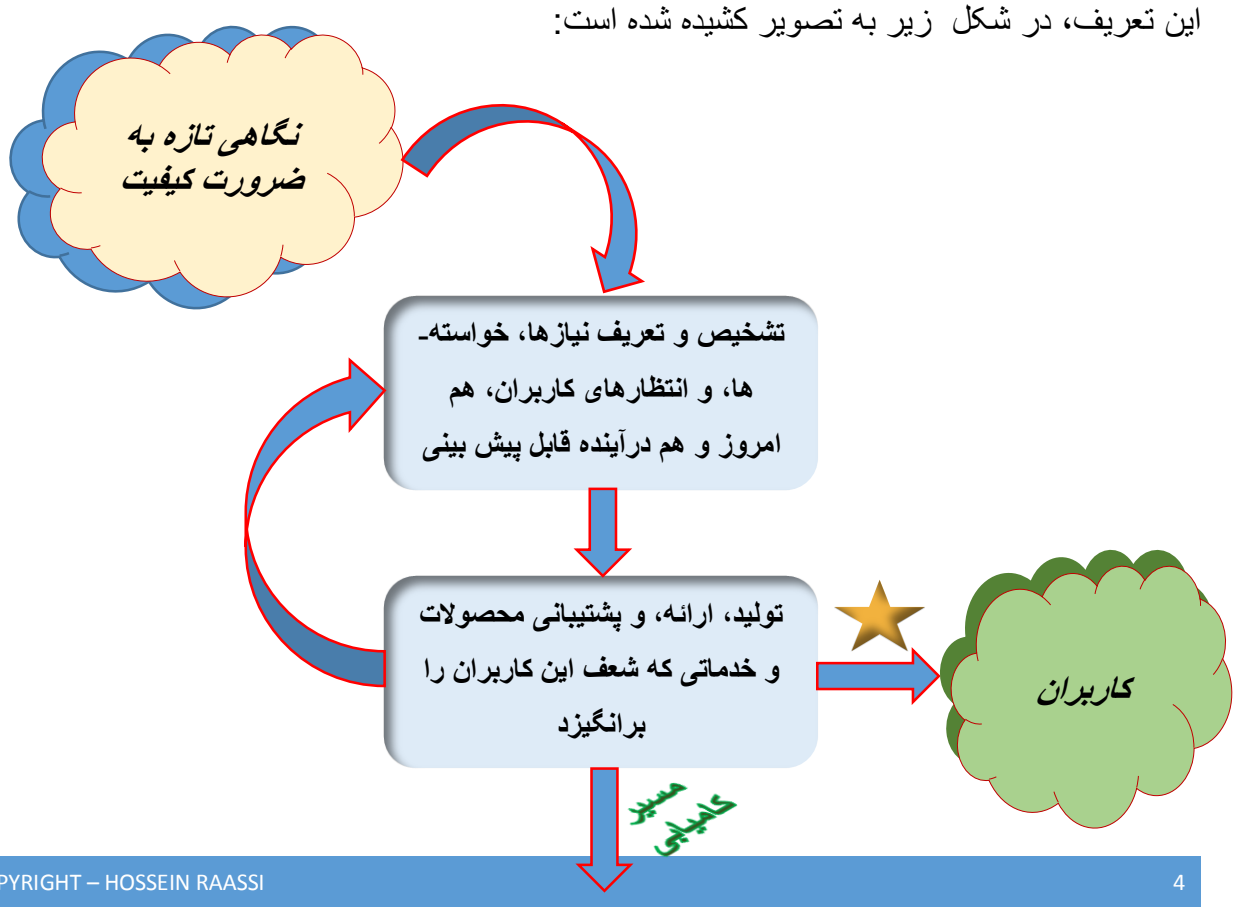
[1] Software Development Life Cycle (SDLC)

SixSigma و نظایر اینها، نیزکوشیده‌اند چارچوب‌ها و تعریف‌های کاربردی و دقیقی از این عبارت ارائه بدهند. از دیدگاه علمی، کیفیت یک محصول (یا سرویس) عبارت است از درجه‌ی انطباق قابلیت‌ها و ویژه گی‌های آن با مشخصات و الزام‌های از پیش تعیین شده. از این دیدگاه، کیفیت در دو بند مجزا، اما مرتبط، تعریف می‌شود: نخست، تحلیل، تدوین، و مشخص سازی الزام‌ها به گونه‌ای صریح، شفاف، و کاملاً تست پذیر؛ و دوم، اندازه گیری درجه‌ی انطباق کارکرد و قابلیت‌های پیدا (و ناپیدای) این محصول (یا سرویس) با الزام‌های تصریح شده در بند نخست.

نگارنده، از بین این تعریف‌ها، یکی را مکرر در صحنه‌ی عمل بکار برده، تأثیرش را به دقت سنجیده، و مضمون آنرا با شرایط روز تطبیق داده است. شفاف ترین بیان این تعریف را می‌توان درسه جمله‌ی زیر خلاصه کرد:

1. تشخیص وتعریف نیازها، خواسته‌ها، و انتظارهای کاربران، هم امروز و هم درآینده قابل پیش-بینی؛
2. تولید، ارائه، و پشتیبانی محصولات و خدماتی که شعف این کاربران را برانگیزد؛
3. تکرار 1 و 2 .

این تعریف، در شکل زیر به تصویر کشیده شده است:



روشن است که هدف و محوراصلی دراین تعریف، «کاربران» هستند، و این انتخاب کاملاً عمدی است، چرا که در تحلیل نهایی، درجه‌ی رضایتمندی همین کاربران (یا مشتریان) است که میزان کیفی محصولات و خدمات ما را اندازه گیری و تعیین می‌کند. نکته‌ی دیگری که دراین تعریف به وضوح بیان شده، برانگیختن وجد و شعف کاربران است. انتخاب واژه «شعف» نیزعمدی است و تأکیدی براین واقعیت که امروز، توقع کاربران ما به مراتب از حد «رضایت» فراتر رفته است؛ آنها انتظار دارند محصولات و خدمات ما آنها را مشعوف کنند. تجربه نشان می‌دهد که کاربران مشعوف، معمولاً شعف شان را با شوق و هیجان با دیگران به اشتراک می‌گذارند، و نتیجه‌ی این تعامل، بازگشت این کاربران و نیز اضافه شدن کاربران جدید به صف مشتریان وفادار است که می‌تواند رونق بیشتر کسب و کار را به دنبال داشته باشد.

لازمه رسیدن به کیفیت مطلوب درمحصولات و خدمات، ایجاد زیرساخت منسجمی است که بتواند با خلق فضایی مناسب برای تحقیق و توسعه‌ی نرم‌افزار، «کیفی اندیشیدن» را، در کنار نوآوری فنی، به یک عادت همیشگی و پایدار تبدیل کند. مرکزیت وهسته‌ی اصلی این زیرساخت را ابزار، تشکیلات، و فرآیندهایی تشکیل می‌دهند که قصدشان هدایت منابع سازمان، به گونه‌ای کارآمد و مؤثر، درمسیرسه قابلیت محوری، اما هم‌پیوند، درفرآیند تولید نرم‌افزاراست. این قابلیت‌ها عبارت از:

همخوانی و سازگاری[2]: یعنی ظرفیت وتوان حفظ نوعی هماهنگی و پیوستگی درتولید و عرضه‌ی (release) محصولات و خدمات به گونه‌ای که حرکت تکاملی فرآیند تولید، پرسش‌های غیرقابل پیش-بینی به میان نیآورد و کاربران را دچار تعجب و حیرت، سرگردانی و گیجی، و یا استیصال و بی-اعتمادی نکند. یکی از راهکارهای مؤثر حفظ این پیوستگی، تعامل نزدیک، مستمر، و پویشگرانه با کاربران بالفعل و بالقوه، در یکایک مراحل فرآیند طراحی و تولید است. درجریان این تعامل، ابهام‌ها زدوده می‌شوند و نیازها و انتظارات کاربران به صورتی شفاف تر و در قالب فیچرهای قابل تست (testable features) به گروه‌های طراحی و تولید منتقل می‌گردند. چالش اصلی در این حوزه، «مدیریت (یا پایش) محتوای» release[3] است.

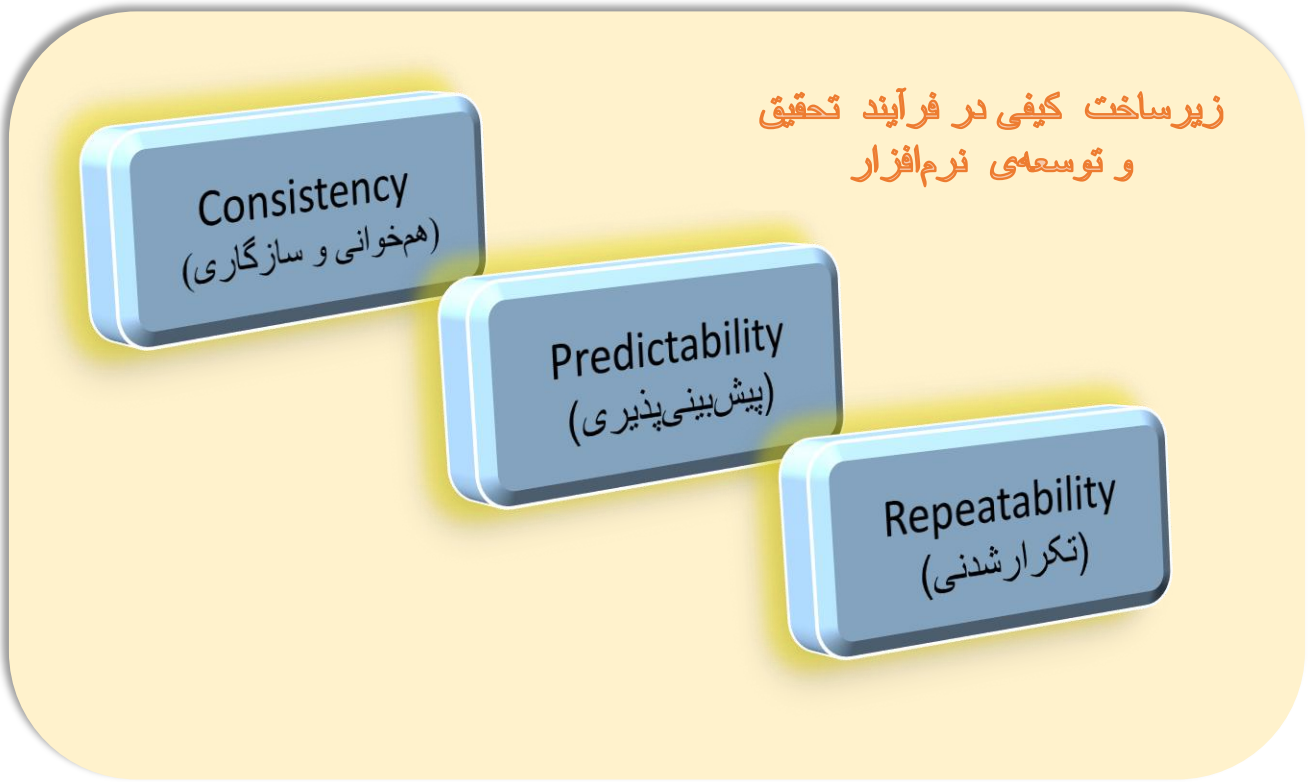
[2] Consistency
[3] Content Management

پیش‌بینی‌پذیری[4]: یعنی ظرفیت و توان پیش‌بینی خروجی و برآمدِ فرایند تولید در هرچرخه، و توان بهینه سازی گام به گام این پیش‌بینی‌پذیری. به زبان ساده تر، این امکان باید برای مدیران طرح‌های تولیدی نرم‌افزار فراهم باشد که آنها بتوانند، به کمک ابزار مربوط و بکارگیری آموخته‌هایی که از تجربه‌های گذشته فراگرفته‌اند، هربارد‌قیق تر از نوبت پیشین، زمان لازم و منابع مورد نیاز (هم انسانی و هم مالی) را برای تولید نرم‌افزاری با الزامات، مشخصات، و ظوابط کیفی از پیش تعیین شده، تخمین بزنند. چالش اصلی در این حوزه، ظهورتغییرهایی است که بعضاً غیرقابل پیش‌بینی بوده و مدیریت آنها[5]، درهرمورد، مهارت و تجربه‌ی خاص می‌طلبد.

تکرارشدنی[6]: یعنی ظرفیت و توان تکرار هرآنچه که در چرخه‌های پیشین طراحی و تولید به برآمد مطلوب رسیده باشد. لازمه‌ی این رویکرد، وجود سازوکاری است که ارزیابی سودمندی از عمکرد، رفتار، و عادت‌های تیم تولید را به همراه رویه‌ها و تکنیک‌های بکار رفته در هر چرخه‌ی تولید به شکلی روشن و قابل تکرار به دست بدهد. بخشی از این ارزیابی را یک یادداشت انتقادی از «مدیریت release»[7]، چالش‌ها، راهکارهای بکار گرفته شده در هر مورد، و میزان توفیق یا ناکامی این راهکارها، تشکیل می‌دهد. در این یادداشت، فهرستی از «آموخته‌های تجربی»[8] نیز که در دوره حیات این چرخه به دست آمده‌اند، به ثبت می‌رسد.

[4] Predictability
[5] Change Management
[6] Repeatability
[7] Release Management
[8] Lessons Learned

شکل زیر، مؤلفه‌های اصلی زیرساخت کیفی مورد بحث را به تصویر می‌کشد:



بدون چنین زیرساخت جاافتاده کیفی، فرآیند تولید به سرعت به چرخه‌های پُرهزینه و پُردردسر آزمون و خطا تبدیل می‌شود و آشفتگی به همه‌ی مراحل آن سرایت می‌کند. درچنین فضایی، کارآیی‌ی تیم تولید کاهش می‌یابد، و طبیعی است که کیفیت محصول هم به تبع آن، درمسیرنامطلوب حرکت خواهد کرد.

همان طورکه در بالا اشاره شد، این زیرساخت درسه زمینه‌ی کلیدی، توان، مهارت، و ظرفیت تیم تولید را به چالش می‌کشد. این سه زمینه عبارتند از:

- مدیریت محتوای release
- مدیریت تغییر
- مدیریت فرآیند release

مدیریت محتوای release، یعنی شناسایی و مشخص سازی مجموعه‌ای بهینه از قابلیت‌های functional و non-functional که در این release ارائه خواهد شد. در بیشترشرکت‌های تحقیق و توسعه‌ی نرم‌افزار، این مسئولیت به عهده تیم «مدیریت محصول»[9] قرار می‌گیرد.

مدیریت تغییر، یعنی توان پذیرش، اولویت بندی، و لحاظ درخواست‌های (عمدتاً غیرقابل پیش‌بینی) در جریان تولید نرم‌افزار. هریک از این درخواست‌ها، پس از بررسی دقیق، ممکن است یا مورد قبول واقع شود، و یا رَد شود. بدیهی است این تصمیم بر اساس اولویت، پیچیدگی، و دامنه‌ی تأثیرهر درخواست بر هزینه و زمان بندی release گرفته می‌شود. مسئولیت مدیریت تغییرها معمولاً به عهده «کمیته کنترل تغییر»[10] قرار می‌گیرد.

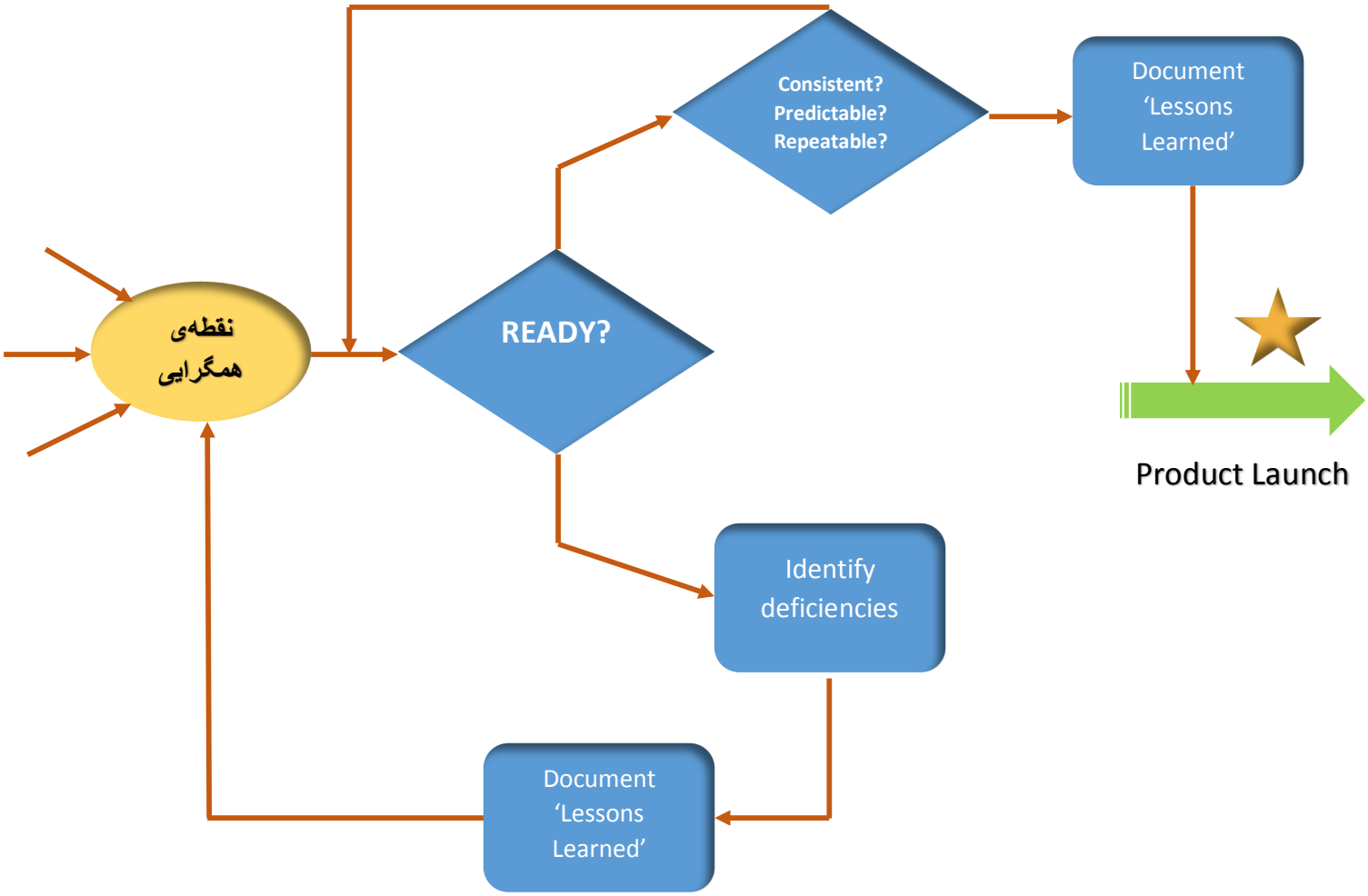
مدیریت فرآیند release، یعنی هماهنگ سازی و ارکستراسیون کلیه‌ی فعالیت‌ها تیم‌های درگیر تولید (development, QE and test, service and support, product management, CCB, project management, configuration management, productivity tools) و هدایت آنها به سوی «نقطه‌ی همگرایی»[11] و release موفقیت‌آمیزمحصول است. علاوه براین، مسئولیت تعریف release criteria و اجرای build ها و(نگهداری ابزارمربوطه) نیز به عهده مدیریت فرآیند release است.

پرسش کلیدی در فرآیند مدیریت release را می‌توان به صورت زیر خلاصه کرد:

- آیا فرآیند تولید به نقطه‌ی همگرایی رسیده و محصول آماده release است؟
- اگر آری، از کجا می‌دانیم چنین شده است؟
  - آیا اصول همخوانی و سازگاری، پیش بینی پذیری، و تکرا رشدنی، در فرآیند تولید محصول رعایت شده اند؟
    - اگر آری، آموخته‌های تجربی مستند سازی شوند، محصول آماده release است.
    - اگر خیر، محصول آماده release نیست.
- اگر خیر، فاصله ما تا نقطه‌ی همگرایی و release محصول چقدر است؟

[9] Product Management (PM)
[10] Change Control Board (CCB)
[11] Convergence Point

شکل زیر، اجزاء این پرسش کلیدی را نمایش می‌دهد:



3. تضمین و استمرار کیفیت در نرم‌افزار

تضمین و استمرار کیفیت در نرم‌افزار، به مثابه فرایندی فراگیر که همه مراحل تحقیق و توسعه را در بر می‌گیرد، نیازمند معماری و برنامه ریزی دقیق، سرمایه گذاری متعهد، دیسیپلین خلل ناپذیر، و مدیریت اجرایی پیگیر و موشکافانه است. تجربه نشان می‌دهد که شرکت‌های نرم‌افزاری که تضمین و استمرار کیفیت در محصولات و خدمات خود را نخستین اولویت فرآیند تولید به حساب می‌آورند، در واقع راه را برای رسیدن به «بلوغ سازمانی»[12] هموار می‌کنند و، برخلاف تصور نادرست عام، مجموع هزینه‌های تولید و پشتیبانی خود را کاهش می‌دهند.

قصد ما در این بخش از مقاله این نیست که تکنیک‌های رایج مهندسی و تضمین کیفیت نرم‌افزار را بازبینی و مرور کنیم، و یا توصیه‌هایی دراین راستا ارائه بدهیم. «برترین روال‌های»[13] مهندسی نرم-افزار، به همراه شاخه‌های مربوط به کیفیت آن، به خوبی در public domain مستندسازی شده‌اند و روش‌های عملی و نظری جدید، دائماً به این مجموعه‌ی پُرارزش افزوده می‌شوند و مورد نقد و بررسی قرار می‌گیرند. اینجا، نیت ما گشودن فصل تازه‌ای در«گفتمان کیفیت» است که به نمودهایی از تضمین و استمرار کیفیت بپردازد که کمتر مورد توجه قرارگرفته‌اند. در پیش‌زمینه‌ی این گفتمان، ما امروز شاهدیم که حتا با وجود برترین روال‌ها، برترین ابزارها، و برترین فرآیندها، تضمین و استمرار کیفیت در محصولات و خدمات، هنوزهم دغدغه‌ی نگران کننده آن گروه از شرکت‌هایی است که در جستجوی انسجام و بلوغ سازمانی، تکاپو می‌کنند. این نمودهایی که، درعین حیاتی بودن، مورد بی توجهی قرار گرفته‌اند کدامند؟

یکی از رایج ترین پنداشت‌های نادرست درمیان تولیدکنندگان نرم افزار (و نیز کاربران)، با این پرسش آغاز می‌شود که: "کیفیت، مسئولیت و وظیفه‌ی کیست؟" و هرپاسخی بجزاینکه: "کیفیت، یک مسئولیت و وظیفه‌ی همگانی است، و همه‌ی دست اندرکاران و ذینفعان، چه درداخل سازمان و چه دربیرون از آن، به نوعی در بود، نبود، ویا کاستی آن سهیم هستند،" به این استنباط غلط دامن می‌زند. کیفیت، مسئولیت و حوزه اختیار یک یا چند نفرنیست، بلکه یک مسئولیت دستجمعی است. در فرآیند تولید نرم‌افزار، تفکر کیفی (یا کیفیتی اندیشیدن)، از لحظه‌ای که تحلیلگران کارِ شناسایی و «مشخص‌سازی الزامات

[12] Corporate Maturity
[13] Best Practices

محصول»[14] را آغاز می‌کنند، شروع شده و بعداً نیز در جریان طراحی و کدسازی و تست و release و پشتیبانی، باید به طور جدی در دستور کار باقی بماند. در پرتو این تفکرکیفی، کیفیت در سرتاسر فرایند تولید، آگاهانه در تار و پود محصول گره می‌خورد و قابل تفکیک از آن نیست. به یاد داشته باشیم که کیفیت مقوله‌ای نیست که بتوان آنرا به دلخواه به اینجا و آنجای محصول تولید شده "تزریق" کرد. طرح-های ضربتی "ترمیم کیفی"، "ارتقاء کیفی"، "بهبود کیفی"، و نظائر اینها، که مدیران ارشد سازمان‌ها درهنگامه‌ی «بحران‌های کیفی»[15] متولی آنها می‌شوند، بیشتر به "پانسمان" شبیه‌اند و غالباً با صرف هزینه‌های سنگین و غیرقابل توجیه، به بیراهه می‌روند و با شکست روبرو می‌شوند. منطقی ترین (و کم هزینه ترین) راه کیفیت بخشیدن به یک محصول بی‌کیفیت، نخست بازاندیشی کیفی آن، و سپس بازتولید آن از نو است، به گونه‌ای که این بار کیفیت مطلوب، در یکایک مراحل فرآیند تولید، در بطن محصول ساخته و پرداخته شود.

یکی از مشخصه‌های اصلی و قابل رؤیت شرکت‌های تولید نرم‌افزار که به درجه‌ای از بلوغ سازمانی دست پیدا می‌کنند، حضورپذیرفته شده همین تفکر کیفی درهمه‌ی سطوح سازمان است. این تفکرکیفی، بخش جداناپذیری از«فرهنگ برتر»[16] است که در این شرکت‌ها فضا را برای نوآوری و شکوفایی خلاقیت‌ها باز نگه می‌دارد.

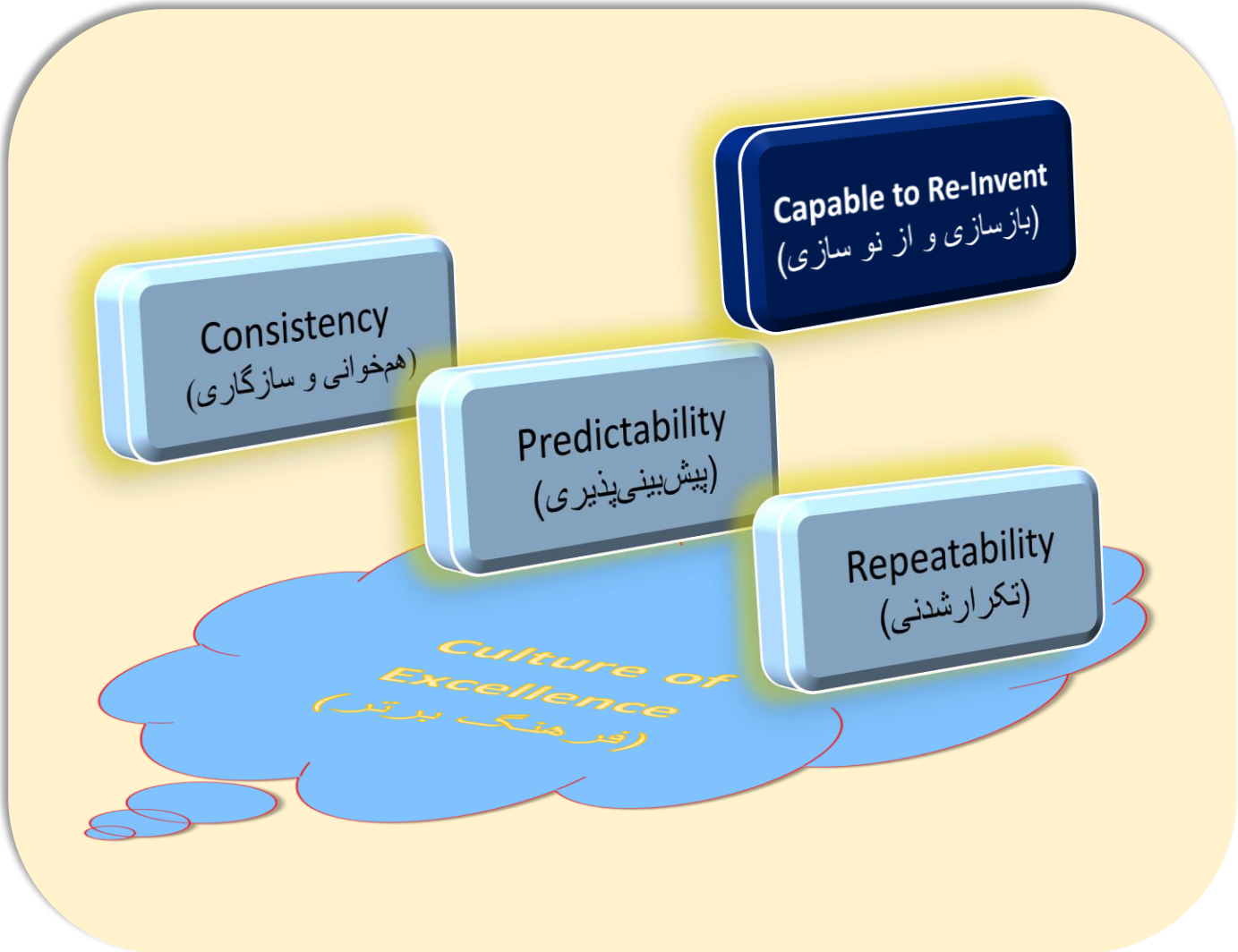
درشرکت‌هایی که بستر«فرهنگ برتر» در آنها شکل گرفته است، کار بی کیفیت (یا کم کیفیت) به هرشکلی و به دست هرکسی، مردود است و کنار گذاشته می‌شود. هزینه‌ها با دقت کنترل می‌شوند و کسی عمداً (یا سهواً) فرصت دوباره کاری و بیهوده کاری پیدا نمی‌کند. به تجربه‌های نو (حتی اگر با شکست مواجه شوند) بها داده می‌شود و کارکنان در همه سطوح به بهره وری و کارآیی بهینه ترغیب می‌شوند. حس پویایی و داوری فنی (و غیر آن) در افراد تقویت می‌شود، و توان و میل به «بازسازی و از نو سازی»[17] در شرکت زنده و بیدار است. در فرهنگ برتر، این توافق عام مستتر است که همه انسان‌ها خطا می‌کنند، پس پذیرش خطا و پوزش به خاطرآن، نشانه ضعف یا گناه نیست، بلکه نمایش اعتماد به نفس و اتکاء پذیری شخصیت انسان است.

[14] Product Requirements Specification
[15] Quality Crises
[16] Culture of Excellence
[17] Capable to Re-invent

تمرکز در پنج حوزه مشخص، استحکام و تداوم فرهنگ برتر را، به عنوان زیرساختی حیاتی درفرآیند تولید نرم‌افزار، تضمین کرده و راه را برای پایداری و بلوغ سازمانی هموار می‌سازد. این پنج حوزه عبارتند از:

- نوآوری مبتنی بر دانش روز، تجربه‌های موفق، و آخرین پیشرفت‌های فناوری
- تقویت، پرورش، و اشاعه‌ی «تفکرکیفی» در بستر «فرهنگ برتر»، درهمه سطوح سازمان
- برانگیختن شوق و شعف در کاربران با محصولات و خدمات شاخص و برتر
- پرورش و توسعه‌ی نیروی انسانی ماهر، کارآمد، و وفادار به ارزش‌های بنیادی شرکت
- کسب اعتبار فزاینده، رقابت‌پذیری، و پیش‌تازی و پیش‌رانی در بازار

شکل زیر، زیرساخت کیفی لازم برای حرکت به سوی بلوغ سازمانی را نمایش می‌دهد:



4. مهندسی کیفیت و تحقق بلوغ سازمانی

دوران حیات شرکت‌های تحقیق و توسعه‌ی نرم‌افزار را می‌توان در سه مرحله، به شرح زیر، مورد بررسی و مطالعه قرارداد:

1. مرحله «تولد، آزمون، نفوذ، و رشد اولیه»[18]
2. مرحله «ادامه رشد، گسترش نفوذ در بازار، انسجام»[19]
3. مرحله «بلوغ و پایداری»[20]

ویژگی‌های اصلی مرحله اول عبارتند از: نوآوری و خلاقیت چشمگیر، حضورمتعهدانه تیم مجرب تحقیق و توسعه، ad hoc، سرعت و چابکی، یک یا دو visionary، موفقیت‌های اولیه، کمینه ساختارسازمانی، کشف و تسخیر سهمی از بازار، جلب اعتماد و وفاداری مشتریان، وهیجان کارکنان و تمایل آنها به کار زیاد و پذیرش مسئولیت‌های گوناگون و تعریف نشده.

ویژگی‌های اصلی مرحله دوم عبارتند از: رشد و توسعه‌ی سریع منابع انسانی، تنوع محصولات و خدمات، گسترش نفوذ در بازار، کنترل و مدیریت پروژه‌ها، انسجام ساختارهای سازمانی (از قبیل بازاریابی، فروش، عملیات، طرح و برنامه، سرویس، منابع انسانی، حسابداری)، استفاده از فرآیندهای تحقیق و توسعه و مدیریت، توان برنامه ریزی و رقابت پذیری.

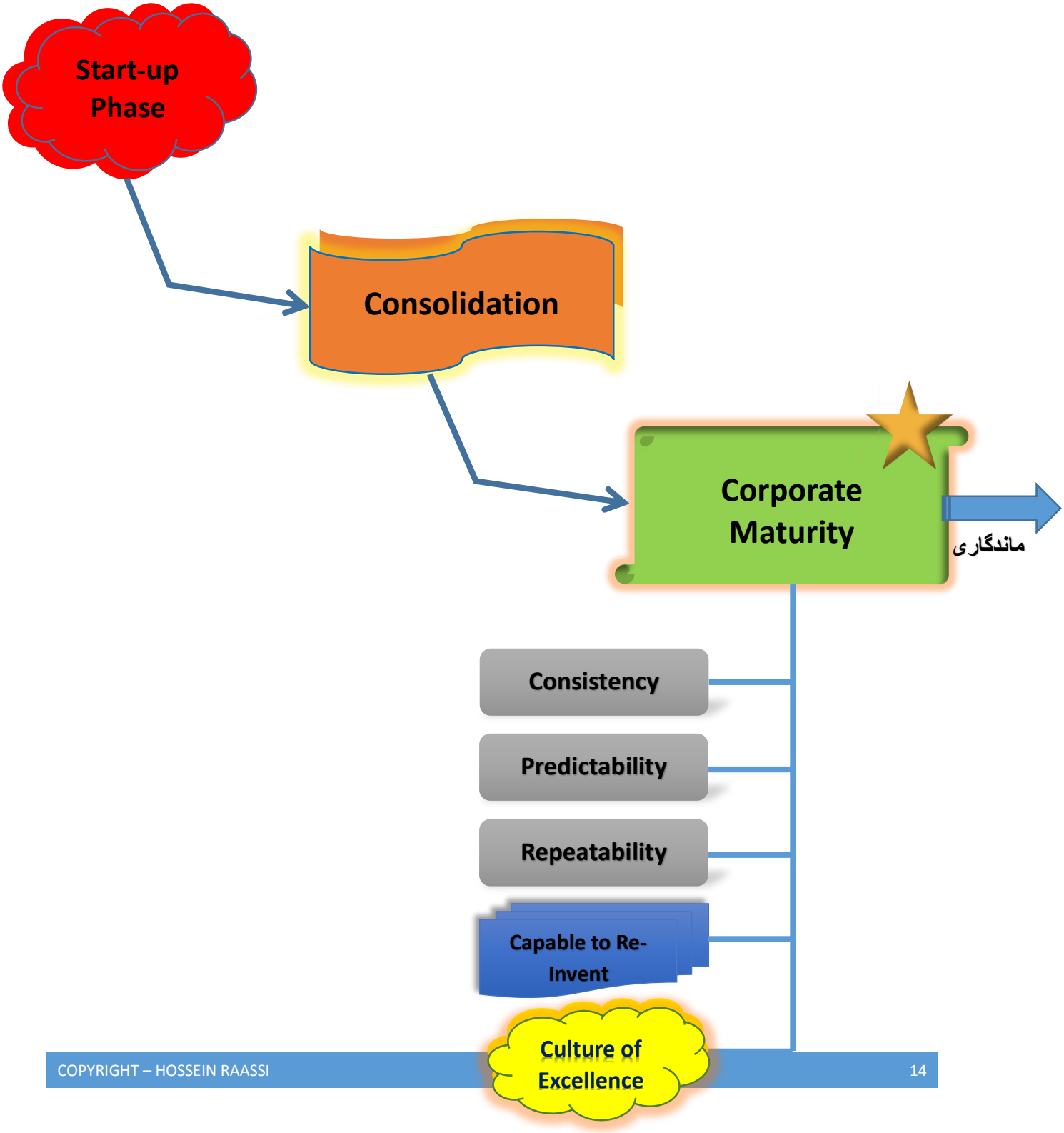
ویژگی‌های اصلی مرحله سوم عبارتند از:

- همخوانی و سازگاری
- پیش‌بینی‌پذیری
- تکرارپذیری
- توان بازسازی و از نو سازی
- بسترمنسجمِ فرهنگ برتر

[18] Startup Phase, Launch
[19] Consolidation Phase, Scale Replication
[20] Corporate Maturity, Culture Transformation

تجربه نشان می‌دهد که گذار از مرحله دوم به مرحله سوم، بدون زیرساخت کیفی – که درفصل پیشین به شرح آن پرداختیم – و نوعی نهادینه‌سازی و تحول فرهنگی در سطح سازمان، امکان پذیرنیست.

شکل زیر، دوران حیات شرکت‌های ماندگار تحقیق و توسعه‌ی نرم‌افزار را به تصویرمی‌کشد:



5. هزینه «فقرکیفی» درنرم‌افزار

مهندسی کیفیت درتولید نرم‌افزار، یعنی نظارت دقیق برجنبه‌های کیفی محصولِ، درهمه مراحل فرآیند تولید، به گونه‌ای که بتوان تابلوی "بالاترین کیفیت" را، با اطمینان خاطر، برخروجی هریک از مراحل تحقیق و توسعه نصب کرد. به عبارت دیگر، محصولی از کیفیت بهینه برخوردارخواهد بود که فرآیند بکاررفته برای تولید آن، درهمه مراحل، از کیفیت بهینه برخوردار باشد. و باز هم به عبارت ساده تر، بین کیفیت یک محصول و کیفیت فرآیند تولید آن، تناسبی مستقیم برقرار است، و این تناسب به عنوان یک اصل، همواره صادق بوده و به حجم یا پیچیدگی نرم‌افزار، و یا متدولوژی بکاررفته، بستگی ندارد.

مهندسی پیوسته‌ی کیفیت در سرتاسر فرآیند تولید، موجب افزایش سرعت تولید می‌شود. به عبارت دیگر، تولید نرم‌افزارِ با کیفیت، زمان تولید را کاهش می‌دهد، و این یک قاعده کلی درمحاسبه زمان بندی تولید نرم‌افزار است که در ظاهر با عقل سلیم سازگار نیست. دلیل این امر این است که ما زمانی را که در گذشته برای اصلاح خطاها و بازنویسی کد صرف می‌شد، حذف کرده‌ایم. یک برنامه نویس معمولی به طور متوسط روزانه 15 تا 25 خط کد جدید بیشترنمی‌نویسد، و بقیه وقتش را صرف خطایابی[21] می‌کند. مهندسی کیفیت با کوتاه کردن زمان خطایابی، در واقع کل زمان تولید را کاهش می‌دهد.

شکل زیر، توزیع زمانی و بهره وری کدنویسی را در فرآیند تولید نرم‌افزارنشان می‌دهد:

Source: *Code Complete*, by Steve McConnell (2007).

[21] Debugging

5.1. اهمیت متدولوژی

متدولوژی‌های توسعه‌ی نرم‌افزار، از «مدل آبشاری»[22] گرفته تا چارچوب Agile، که طیف گسترده‌ای را دربرمی‌گیرند، به تنهایی و بدون مهندسی کیفیت، کمتربه نتیجه کیفی مطلوب دست پیدا می-کنند. در ذهن گروهی از مدیران و developer های کلیدی نرم‌افزار، این پنداشت نادرست شکل گرفته است که کلید رفع همه کاستی‌های کیفی، در مهاجرت از یک متدولوژی به متدولوژی دیگر نهفته است. اخیراً هم می‌بینیم که طرفداران تکنیک‌های Agile، مثلاً Scrum، خواسته یا ناخواسته، با هیجان به این پنداشت نادرست دامن می‌زنند. درهمین حال، آمار نشان می‌دهد که بیش از پنجاه درصد از تیم‌های توسعه‌ی نرم‌افزارکه خود را Agile می‌انگاشته‌اند، پس ازیک یا چند دوره ناکامی و با صرف هزینه‌های غیرقابل توجیه، دوباره به متدولوژی‌های «سنّتی» (برخاسته از مدل آبشاری و یا تبار iterative آن) روی آورده‌اند. البته هدف ما در اینجا نفی Agile یا هرمتدولوژی دیگر، و یا نفس مهاجرت از یکی به دیگری نیست، چرا که درعمل هریک از این متدولوژی‌ها، کاربردِ بهینه و جایگاه مناسب خود را دارد. اما باید یادآورشویم که انتخاب متدولوژی مناسب برای توسعه‌ی نرم‌افزار، خود نیز تابع نوعی تفکر و فرآیند کیفی است که با توجه به عوامل تعیین کننده‌ای نظیر نوع، حوزه عملکرد، و حجم محصول، کاربران واقعی و بالقوه، و مهارت و تجربه فردی و دستجمعی تیم توسعه، صورت می‌پذیرد. پس ازانتخاب متدولوژی، ضرورت و تداوم همین تفکر و مهندسی کیفی است که درهریک از مراحل فرآیند تحقیق و توسعه، کیفیت مطلوب را در محصول نهایی تضمین می‌کند.

تجربه دهه‌های اخیر در شرکت‌های جهانی و صاحب نام نرم‌افزاری نیز به خوبی شاهد این واقعیت است که اتکاء به متدولوژی، بدون توجه کافی به جنبه‌های کیفی فرآیند تولید، غالباً به تولید و عرضه محصولاتی منجرشده است که دچار «فقر» ذاتی کیفیت هستند و به همین دلیل شعفی در کاربران برنمی-انگیزند. ریشه‌های اصلی این فقر کیفی را باید، بیش از همه، در مراحل نخستین فرآیند تولید محصول و پیش از شروع تستِ functional، یعنی تشخیص وتحلیل الزامات[23]، طراحی مفهومی[24]، توسعه‌ی معماری[25]، طراحی درسطح جزییات[26]، و سپس کدسازی[27] (برنامه نویسی) جستجو کرد. در این مراحل

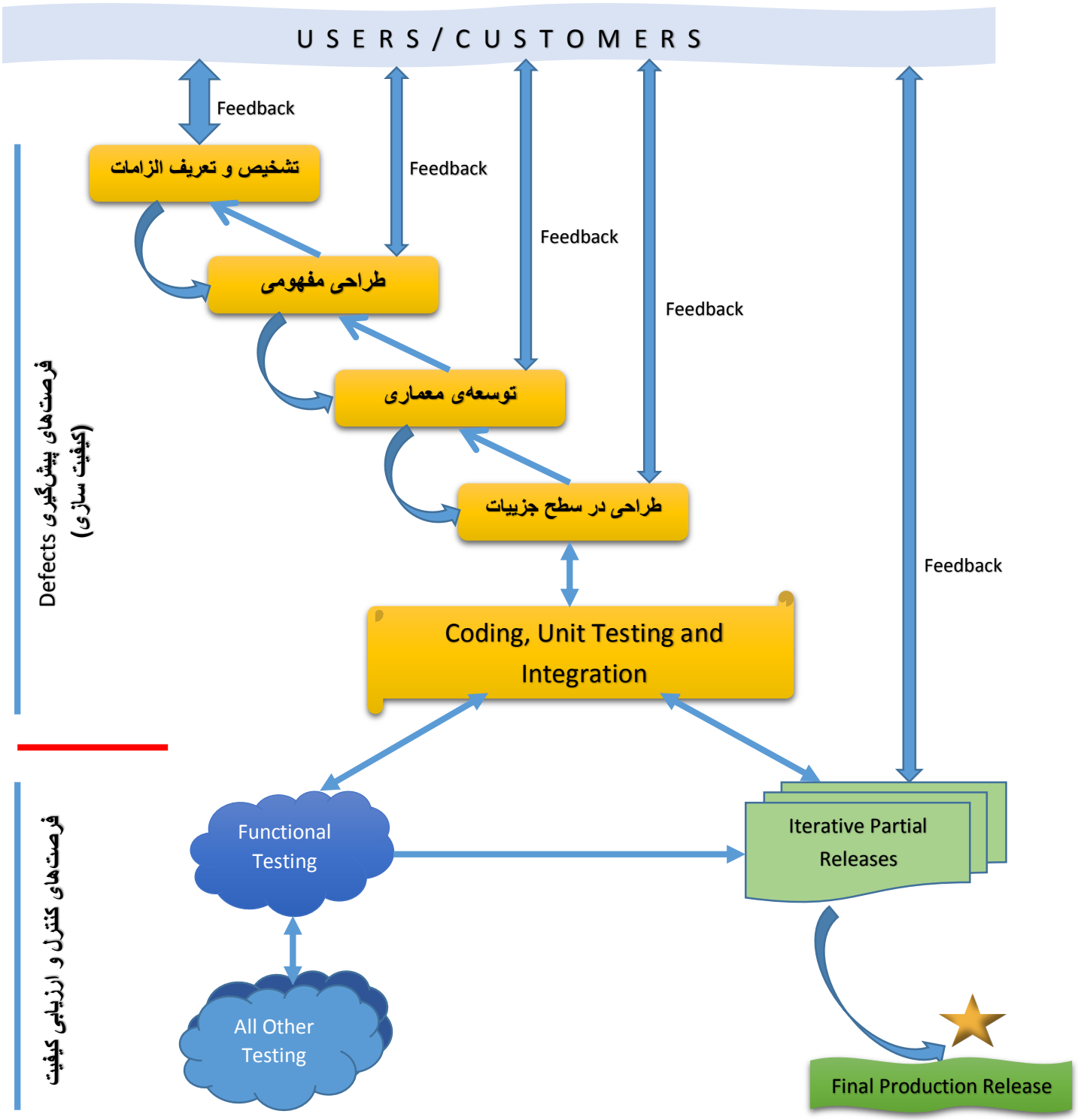
[22] Water Model
[23] Requirements Analysis
[24] Conceptual Design
[25] Architecture Development
[26] Detailed Design
[27] Coding (Code Construction/Programming)

است که فرصت اعمال تفکرکیفی برای پیشگیری defect ها، یعنی «کیفیت سازی»، بیش از مراحل بعدی در اختیار تیم توسعه قرار می‌گیرد. با شروع مرحله تستِ functional، فرصت‌های پیش‌گیری از دست رفته‌اند و از این پس کار به نوعی «کنترل کیفیت»[28]، محدود می‌شود. در این مرحله، تمرکز نیروها بر کاهش defect های باقیمانده است، نه پیشگیری از تولید آنها. هرچه که پیچیدگی محصولات و خدمات بیشتر شود، ضرورت این گونه تفکرکیفی نیز بیشتر می‌شود.

به طور کلی، تکنیک‌ها، ابزار، و رویه‌های مهندسی کیفیت نرم‌افزار به شکلی تکامل یافته‌اند که انتخاب متدولوژی تولید، تأثیر کمی بر کارآیی و نتایج مورد انتظار دارد.

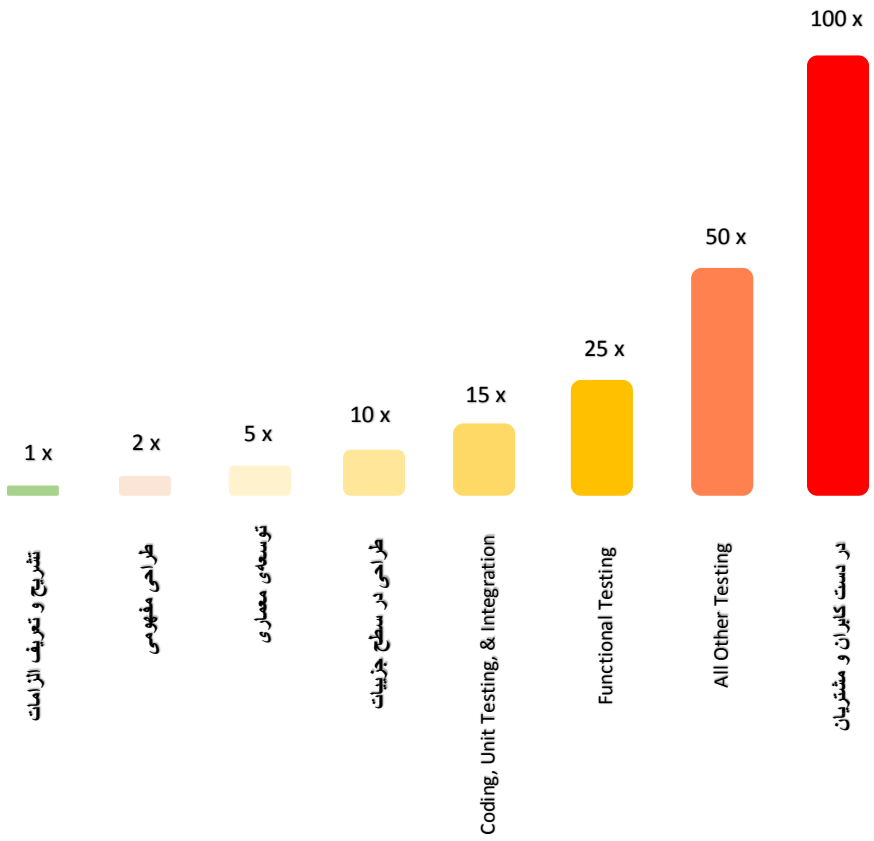
درتصویر زیر، مراحل مشخص فرآیند توسعه و فرصت‌های پیشگیری و کاهش defect ها، در قالب یک مدل iterative نشان داده شده‌اند:

[28] Quality Control


USERS / CUSTOMERS
Feedback
تشخیص و تعریف الزامات
Feedback
طراحی مفهومی
Feedback
توسعه‌ی معماری
Feedback
طراحی در سطح جزییات
Feedback
Coding, Unit Testing and Integration
فرصت‌های پیش‌گیری Defects
(کیفیت سازی)
Functional Testing
All Other Testing
Iterative Partial Releases
Final Production Release
فرصت‌های کنترل و ارزیابی کیفیت

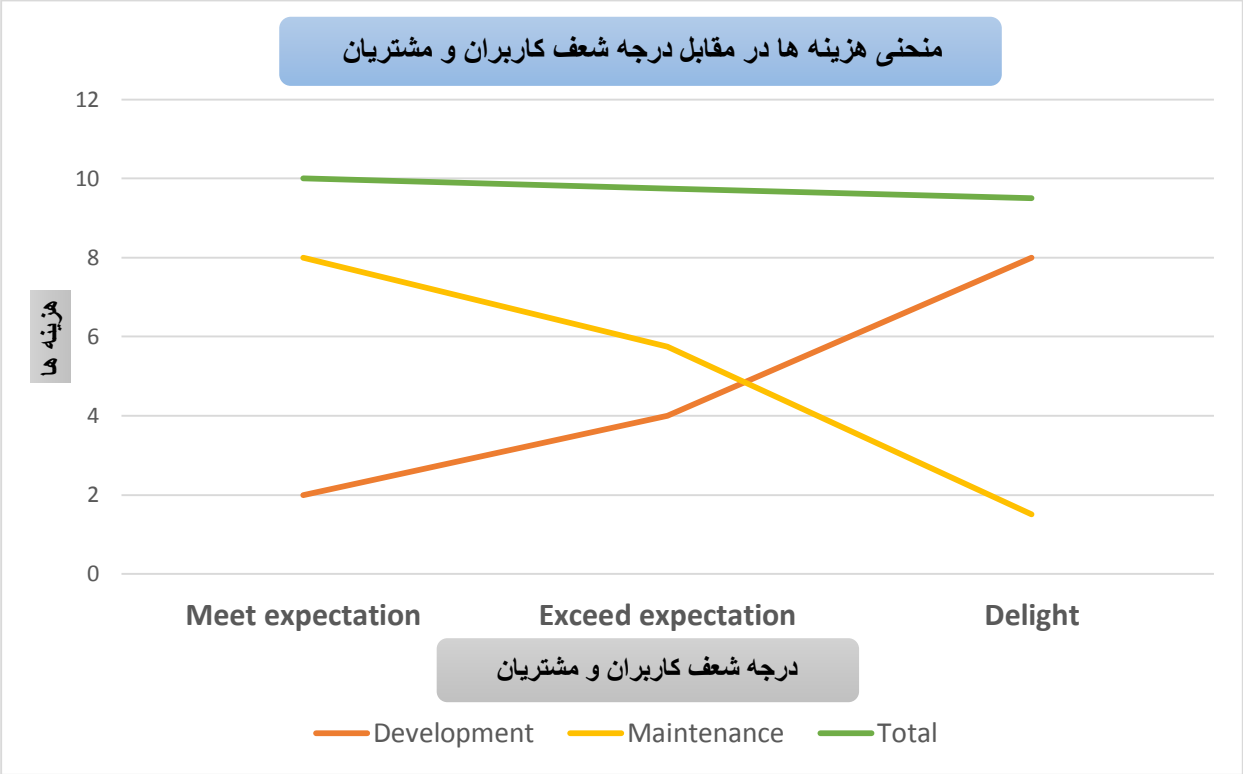
5.2. هزینه اصلاح، تعمیر، و یا تصحیح defect ها

در فرآیند تحقیق و توسعه‌ی نرم‌افزار، هرچه که حیات یک عیب یا نقص، یک کاستی، یا یک خطا[29]، ادامه پیدا کند، اصلاح یا تصحیح آن سخت تر است و گران تر تمام می‌شود. درفرآیند تولید محصولات بزرگ، که غالباً شامل صدها یا هزاران module و دهها یا صدها هزار خط کد هستند، هزینه اصلاح defect یی که اصلاح نشده از یک مرحله به مرحله بعدی فرآیند تولید منتقل شود، بین 5 تا 25 برابر می‌شود. به عنوان مثال، اصلاح defect ی که در مرحله «طراحی مفهومی» تولید شود، اما کشف و اصلاح آن تا مرحله «تست» به تعویق بیافتد، ممکن است تا 50 برابر هزینه کشف و اصلاح آن در همان مرحله طراحی مفهومی افزایش یابد. از این گذشته، آمار نشان می‌دهد که در مواردی که فقر کیفی گریبانگیر فرآیند تولید است، developer های کلیدی تیم‌های توسعه‌ی نرم‌افزار متجاوز از 50 درصد وقت مفید خود را صرف خطایابی و سپس اصلاح، تعمیر، و یا تصحیح همین خطاها می‌کنند. شکل زیر نمودار رشد تصاعدی هزینه‌ی تقریبی اصلاح، تعمیر، و یا تصحیح defect هایی را نشان می‌دهد که کشف آنها در فرآیند تحقیق و توسعه به تعویق می‌افتد:



[29] Defect

به این ترتیب نقش مؤثر، و ضرورت تفکر و مهندسی کیفیت در کاهش هزینه‌های فرآیند تحقیق و توسعه-ی نرم‌افزار آشکار می‌شود، و نمودار زیر تصویری از این نتیجه گیری است:

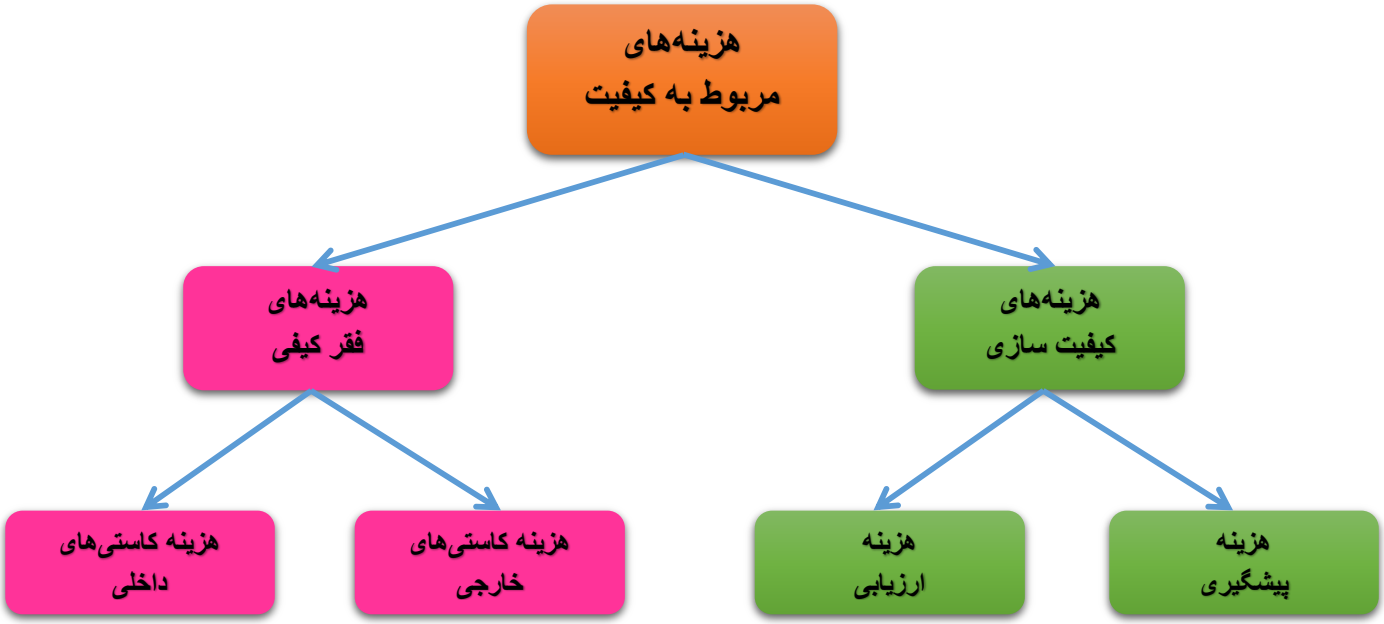


5.3 هزینه‌های تفکیف شده کیفیت

در گفتمان کیفیت نرم‌افزار که امروزه هم در محافل آکادمیک و هم در صحنه عمل رایج است، هزینه‌های مربوط به کیفیت در دو گروه قابل بررسی و اندازه گیری هستند:

1. هزینه‌های مربوط به کیفیت سازی، شامل
   - الف. هزینه پیش‌گیری
   - ب. هزینه ارزیابی
2. هزینه‌های ناشی از فقرکیفی، شامل
   - الف. هزینه کاستی‌های داخلی
   - ب. هزینه کاستی‌های خارجی

شکل زیر، دسته بندی هزینه‌های مربوط به کیفیت را نشان می‌دهد:



هزینه کاستی‌های داخلی، عمدتاً هزینه کشف و اصلاح defect هایی است که قبل از release محصول به تیم تحقیق و توسعه تحمیل می‌شود. این هزینه‌ها شامل ترمیم و تصحیح خطاها در شرح الزامات و معماری‌ها و طراحی‌های قبل از کدسازی؛ مرور و اعمال این اصلاحات؛ کشف، گزارش، تصحیح، و تست defect ها؛ چرخه‌های integration و تست (functional, non-functional, regression)؛ build ها و نسخه‌سازی‌های خارج از برنامه زمان بندی؛ و تأخیر در زمان release؛ می‌شوند.

هزینه کاستی‌های خارجی، عمدتاً هزینه کشف و اصلاح defect هایی است که بعد از release محصول درمحیط کاربران و مشتریان ظاهرمی‌شوند. هزینه‌های مربوط به release های خارج از برنامه زمان بندی شده (minor, patch, hotfixes)، هزینه‌های پشتیانی و نگهداری، هزینه‌های مربوط به جبران خسارت‌ها و مرجوعی‌ها، هزینه‌های حقوقی ناشی از شکایات، مشتریان ناراضی، لغو قراردادها، و کاهش اجباری قیمت‌ها، نیز در فهرست کاستی‌های خارجی قرار می‌گیرند.

هزینه پیشگیری، شامل هزینه‌های مربوط به برنامه ریزی کیفیت، آموزش مهندسی کیفیت، کیفیت سازی و تضمین کیفیت، بکارگیری کارآمد ابزار و فرآیندهای بهره وری (configuration management, defect tracking, code coverage analysis, memory leak detection)، مدیریت تغییرات، چرخه‌های مرور و بازنگری و بازرسی (reviews and inspections) ، و بکارگیری مجدد نرم‌افزار-هایی که قبلاً مورد استفاده قرارگرفته‌اند (software reuse)، می‌شود.

هزینه ارزیابی (یا کنترل کیفیت)، عمدتاً هزینه تعیین درجه تطبیق نرم‌افزار با الزامات مشخص و تعریف شده آن است. اندازه گیری این درجه‌ی تطبیق، شامل اجرای نخستین iteration تست‌های integration و سیستم، و گزارش defect های کشف شده درجریان اجرای این تست‌ها است. هزینه‌های مربوط به ابزار و فرآیندهای automation تست‌ها نیز بخشی از هزینه ارزیابی را تشکیل می‌دهند.

6. سخن آخر

نخستین گام در بهینه سازی عملکرد مهندسی نرم‌افزار، تشخیص و شناسایی دقیق کاستی‌ها و ضعف-هایی هستند که به فقرکیفی درفرآیند تحقیق و توسعه‌ی نرم‌افزارهم‌افزایی می‌کنند. با شناخت این کاستی‌ها و ضعف‌ها، می‌توان تفکرکیفی را در بطن فرآیندهای مهندسی نرم‌افزار نهادینه کرد. کوشش ما در این مقاله این است که توجه خواننده را به نمودهای کمترشفاف «گفتمان کیفیت نرم‌افزار» جلب کرده و در عین حال نشان دهیم که همزمان با کاهش defect ها درفرآیند تحقیق و توسعه، هزینه جمعی تولید نیز کاهش می‌یابد، و این درحالی است که کیفیت محصول به همراه بهره وری تیم تولید، به گونه قابل ملاحظه‌ای افزایش پیدا می‌کند.

بسیاری از مدیران و تولید کنندگان نرم‌افزارمی‌پندارند که لازمه‌ی کیفیت برتر، خرج هزینه بیشتر و تخصیص زمان بلند تر، درفرآیند تحقیق و توسعه است. حال آنکه تجربه شرکت‌هایی که، استوار بر فرهنگ برتر و تعهد فزاینده به پیشتازی و پایداری، مهندسی کیفیت را در یکایک مراحل تولید نخستین اولویت قرار می‌دهند، به روشنی نشان می‌دهد که بازدهی‌ی این سرمایه گذاری حیاتی به مراتب از حد انتظارشان فراتر رفته است.

از یاد نبریم که کیفیت محصولات و خدمات ما تابعی است از کفایت و کیفیت نیروهای انسانی درگیر و نیز کارآیی و کیفیت فرآیندهای زیرساخت تولید.

7. منابع و مآخذ

1. ASQ: The Global Voice of Quality
   http://asq.org/pub/qe/
2. *Better Software Magazine*
   http://www.sqe.com/Publications/ and http://www.stickyminds.com/
3. Tian, J., *Software Quality Engineering: Testing, Quality Assurance, and Quantifiable Improvement*, 2005.
4. Kan, Steven, H., *Metrics and Models in Software Quality Engineering*, 2003.
5. Jones, C., and Bonsignour, O., *The Economics of Software Quality*, 2011.
6. Borror, Connie M. *The Certified Quality Engineer Handbook, Third Edition*, 2008.
7. Defeo, J., and Juran, J. M., *Juran's Quality Handbook: The Complete Guide to Performance Excellence 6/e*, 2010.
8. Salinesi, C., and Van De Weerd, I., *Requirements Engineering: Foundation for Software Quality: 20th International Working Conference*, REFSQ 2014.
9. Bass, L., Clements, P., and Kazman, R., *Software Architecture in Practice (3rd Edition)* (SEI Series in Software Engineering), 2012.
10. Linz, T., *Testing in Scrum: A Guide for Software Quality Assurance in the Agile World*, 2014.
11. Whittaker, James, A., *How Google Tests Software*, 2012.
12. Sommerville, I., *Software Engineering (9th Edition)*, 2010.
13. Fowler, C., *The Passionate Programmer: Creating a Remarkable Career in Software Development (Pragmatic Life)*, 2009.
14. Pressman, R. and Maxim, B., *Software Engineering: A Practitioner's Approach*, 2014.

15. Chemuturi, M., *Mastering Software Quality Assurance: Best Practices, Tools and Techniques for Software Developers*, 2010.
16. Westfall, L., *The Certified Software Quality Engineer Handbook*, 2009.
17. McCaffrey, James, D., *Software Testing: Fundamental Principles and Essential Knowledge*, 2009.
18. Schulmeyer, Gordon, G., *Handbook of Software Quality Assurance*, 2007.
19. McConnell, S., *Code Complete: A Practical Handbook of Software Construction, Second Edition*, 2004.
20. Fox, A., and Patterson, D., *Engineering Software as a Service: An Agile Approach Using Cloud Computing*, 2013.
21. Tsui, F., Karam, O., *Essentials Of Software Engineering*, 2013.
22. Braude, Eric. J., and Bernstein, Michael. E., *Software Engineering: Modern Approaches*, 2010.
23. Lawrence, S, et al., *Software Engineering: Theory and Practice*, 2009.
24. ISO/IEC 25010:2011 http://www.iso.org/iso/catalogue_detail.htm?csnumber=35733
25. IEEE Software Engineering Standards http://www.ieee.org/portal/innovate/products/standard/ieee_soft_eng.html
26. Information Technology Infrastructure Library (ITIL) www.itil.org
27. Software Engineering Institute (SEI) – Carnegie Mellon University http://www.sei.cmu.edu/