Specialized Scientific Quarterly Journal of

Arman Process (APJ)

Investigating the quality management dimensions in software architecture

***A. Sobhani*[*,1]**

[1] *Department of Electrical and Computer Engineering, North Tehran Branch, Islamic Azad University, Tehran, Iran*

ABSTRACT

Investigating the quality management dimensions and related requirements of software architecture is one of the most important factors for the success of today's software in the commercial market. This process is associated with fundamental challenges that hinder the development of solutions. Underestimating the extraction of qualitative dimensions of software may cause very costly problems in the final stages of development and cause the analyst to face many problems and obstacles, in this regard. These problems that underlie the present study can be caused by many factors that can be avoided during the process of extracting the quality requirements. The ability of engineering process requirements in the system, such as methods and tools, to deal with changes are very important factors. In today's complex systems, due to the existence of very large data, communication and flow between them and the Internet of Things, very large systems have been provided that according to the scale of these software systems, validation is one of the final requirements to confirm these quality requirements. In this study, we have examined the dimensions of software quality management from the perspective of stakeholders and developers, the validation of quality requirements and also how to use the quality dimensions in the modern software systems due to its great importance in this field.

KEYWORDS:

Quality management
Analyze
Verify
Software quality requirements
Software Architecture

Corresponding author
A_Sobhani@yahoo.com

NUMBER OF REFERENCES	NUMBER OF FIGURES	NUMBER OF TABLES
20	0	0

فصلنامه تخصصی

آرمان پردازش

بررسی ابعاد مدیریت کیفیت در معماری نرم افزار

علی سبحانی*[1]

[1] گروه کامپیوتر، دانشکده برق و کامپیوتر، واحد تهران شمال، دانشگاه آزاد اسلامی، تهران، ایران

چکیده

بررسی و تحلیل ابعاد و نیازمندی های کیفی معماری نرم افزار از مهم ترین عوامل موفقیت نرم افزارهای امروزی در بازار تجاری می باشد. این فرآیند با چالش هایی اساسی همراه است که توسعه راه حل ها را مختل می نماید. کم اهمیت قلمداد کردن استخراج ابعاد کیفی نرم افزار ممکن است در مراحل پایانی توسعه مشکلات بسیار پرهزینه ای را به بار آورده و تحلیلگر را در این مورد با مشکلات و موانع بسیاری مواجه سازد. این مشکلات که زمینه ساز پژوهش حاضر می باشد می تواند ناشی از عوامل بسیاری باشد که شناخت این موارد می تواند در حین فرآیند استخراج نیازمندی ها سبب اجتناب از آن ها گردد. توانایی فرآیندهای مهندسی نیازمندی ها در سیستم از قبیل روش ها و ابزارها، برای مقابله با تحولات از عوامل بسیار مهمی هستند. در سیستم های پیچیده امروزی به دلیل وجود داده های بسیار بزرگ، ارتباطات و جریان بین آنها و اینترنت اشیاء سیستم هایی بسیار عظیم ارائه شده است که با توجه به مقیاس این سیستم ها اعتبارسنجی یکی از الزامات نهایی در جهت تأیید این نیازمندی‌های کیفی است. در این پژوهش برررسی ابعاد مدیریت کیفیت نرم افزار از منظر ذینفعان و سازنده، اعتبارسنجی نیازمندی های کیفی و همچنین چگونگی بهره برداری از ابعاد کیفی در سیستم نرم افزاری را به جهت اهمیت زیاد در این حوزه مورد بررسی قرار داده ایم.

واژگان کلیدی:
مدیریت کیفیت
تجزیه تحلیل
وارسی
نیازمندی‌های کیفی نرم افزار
معماری نرم افزار

[1]نویسنده مسئول

A_Sobhani@yahoo.com

تعداد جداول	تعداد شکل ها	تعداد مراجع
۰	۰	۲۰

مقدمه

با پیشرفت سریع علم مهندسی نرم افزار، تولید نرم‌افزارهای کاربردی روزبه‌روز گسترش می‌یابد و لزوم به کارگیری روش‌ها و اصول مدون و جامع، در مراحل توسعه، مدیریت و پشتیبانی نرم افزارها بیشتر نمود پیدا می‌کند. معماری نرم افزار شاخه ای از مهندسی نرم افزار است که به کمک آن می توان ساخت نرم افزار های بزرگ و پیچیده را ساده نموده، هزینه های تولید و نگهداری را کاهش داده، کیفیت محصولات نرم افزاری را ارتقاء بخشید و زمینه ای مناسب را برای توسعه سیستم های نرم افزاری ایجاد نمود. مدل‌سازی نرم‌افزار، به کارگیری فنون پیشرفته آزمایش نرم‌افزار، مدیریت ریسک نرم‌افزار، تضمین کیفیت نرم‌افزار و مهندسی محصول، تنها عناوینی از فهرست گسترده زیرساخت‌های مرتبط با توسعه نرم‌افزارهای قوی و مهندسی ساز است [۲و۱].

همانطور که قبلاً گفته شده، یکی از اهداف مهم در فرایند تولید سیستم های نرم افزاری، ارزیابی معماری نرم افزار جهت اطمینان از حصول نیازهای سهامداران در سیستم نرم افزاری ساخته شده بر مبنای معماری ارائه شده می باشد، که مهمترین هدف در حوزه ارزیابی معماری نرم افزار، ارزیابی ویژگی های کیفی معماری می باشد. کیفیت نرم‌افزار، شاخص حیاتی برای تولید نرم‌افزارهای با کیفیت بالا است که ضمن بالا بردن بهره‌وری در تولید سیستم های نرم‌افزاری، به ایجاد نرم‌افزارهای قدرتمند و شکست‌ناپذیر منجر می‌گردد. امروزه روش های مختلفی برای توصیف و ارزیابی کیفی معماری ارائه شده است که هدف برخی از آنها این است که برای معمار طراح امکان مقایسه بین معماری های کاندید و انتخاب معماری مناسب از بین آنها را فراهم کند. مشکلی که در ارزیابی ویژگی های کیفی مطرح می باشد، عدم شناخت ابعاد مساله و فقدان یک روش سیستماتیک و رسمی جهت ارزیابی ویژگی های کیفی می باشد [۳]. لذا در این مقاله قصد داریم درخصوص مساله مهندسی ویژگی های کیفی معماری و نرم افزار و ابعاد مختلف این مساله به تفضیل بحث نمائیم. همچنین در این فصل، سعی بر آن است با معرفی مدل‌ها و استانداردهای معروف موجود در حوزه ارزیابی کیفیت معماری و نرم‌افزار، آشنایی نسبی با پژوهش‌های علمی که تا کنون در این حوزه صورت گرفته، به دست آید. شفاف سازی ابعاد این مساله می تواند گامی بسیار مهم درجهت تولید و توسعه سیستم های نرم افزاری با ویژگی های کیفی مطلوب ایجاد نماید.

مشخصه های کیفیتی سیستم های نرم افزاری

امروزه سیستم های کامپیوتری در بسیاری از برنامه‌های رایج و حتی کاربردهای بحرانی بیش از پیش مورد استفاده قرار می‌گیرد. در این برنامه‌ها وجود یک نقص می‌تواند عواقب زیادی را به دنبال داشته باشد. توسعه دهندگان این سیستم ها، مسئول تشخیص و تعیین نیازمندی های این برنامه‌های کاربردی و ایجاد سیستم برای محقق سازی این نیازمندی ها هستند. توسعه دهندگان این سیستمها، ابتدا نیازمندی های برنامه مورد نظر را شناسایی می‌کنند، نرم‌افزار مورد نظر را طوری توسعه می‌دهند که نیازمندی های مورد نظر را با منابع مقتضی پوشش داده شوند. سیستم های بحرانی در حالت کلی نیازمند برخی مشخصه‌های اضافی دیگر نیز هستند که می‌توان به کارایی، وابستگی، امنیت، سلامتی و برخی نیازهای مشابه اشاره کرد [۵و۴].

نیازمندی ها در حالت کلی به دو دسته نیازمندی های عملیاتی و نیازمندی های غیرعملیاتی تقسیم می‌شوند. نیازمندی های عملیاتی، عبارتست از توانایی سیستم در انجام کاری که برای آن ایجاد شده است. نیازمندی های غیرعملیاتی یک سیستم نرم افزاری که تحت عنوان مشخصه‌های کیفی از آنها یاد می‌شود، هر آنچه که غیر از نیازمندی های عملیاتی سیستم باشد، را پوشش می دهد. نیازمندی هائی مانند کارایی، امنیت، هزینه ساخت و سایر موارد مرتبط. کیفیت سیستم نرم‌افزاری به صورت مستقیم با توانایی یک سیستم در قبال نحوه انجام نیازمندی های عملیاتی و غیرعملیاتی آن در ارتباط می‌باشد. یک سیستم می‌تواند شامل مشخصه‌های زیادی همچون کارایی، قابلیت نگهداری، امنیت و... باشد. اساسا کیفیت هر یک از مشخصه‌های موجود، بر کیفیت کل سیستم تاثیر دارد. یعنی کیفیت کل سیستم تابعی از کیفیت تک‌تک مولفه ها و مشخصه‌های جزء می‌باشد. لازم به ذکر است که همیشه کیفیت این مشخصه‌ها به راحتی قابل اندازه‌گیری نیست. کیفیت بالای محصول نرم‌افزاری، به صرفه‌جویی در هزینه و ارتقای همیشگی سطح نرم‌افزار می‌انجامد. تمامی توسعه‌دهندگان نرم‌افزاری توافق دارند که دستیابی به نرم‌افزارهای با کیفیت، بالاترین هدف در ایجاد و ساخت سیستم‌های نرم‌افزاری است؛ اما یک سوال اساسی این است که کیفیت نرم‌افزار چگونه تعریف می‌شود؟ اساسا مدیریت و تضمین کیفیت نرم‌افزار مطابق با نیازهای عملیاتی و استانداردهای توسعه نرم‌افزار تعریف و تدوین می‌گردد و در این میان، توجه به سه اصل زیر اهمیت دارد [۷و۶] :

- استانداردها، مجموعه‌ای از معیارهای توسعه را تعریف می‌کنند و چنانچه این معیارها به درستی دنبال نشوند، نتیجه آن فقدان کیفیت خواهد بود.
- چنانچه یک نرم‌افزار بر نیازهای اصلی خود منطبق باشد، اما نیازهای جانبی خود مانند سهولت کاربری و پشتیبانی مناسب را برآورده نسازد، کیفیت نرم‌افزار حاصل نگردیده است.

- نیازمندی‌های نرم‌افزار و آنچه نرم‌افزار برای آن طراحی و پیاده‌سازی گردیده، مبنای اندازه‌گیری کیفیت است. عدم تطبق نرم‌افزار با نیازمندی‌های آن، موجب عدم کیفیت نرم‌افزار خواهد شد.

در گذشته یک افسانه متداول در توسعه سیستم های نرم‌افزاری این بود که ابتدا نرم‌افزاری ساخته شود که نیازمندیهای وظیفه‌ای را برآورده کند، سپس نیازمندیهای غیروظیفه‌ای به آن افزوده یا تزریق شود. این ایده ما را به سمت اتلاف منابع و کیفیت پایین سوق می‌دهد. لذا، باید تمرکز بر روی کیفیت را از همان ابتدای چرخه حیات سیستم نرم افزاری و درسطح معماری آغاز کرد. وظیفه‌مندی و صفات کیفیتی از نظر تئوری متعامد هستند. لذا، نمی‌توان با هر سطحی از وظیفه‌مندی به همه صفات کیفیتی دست یافت. وظیفه‌مندی را می‌توان به شکل‌های مختلف بدست آورد و چندان به معماری مربوط نیست. اما عموما، معماری قصد دارد با سازماندهی وظایف به صفات کیفیتی و غیرعملیاتی دست پیدا کند. دستیابی به کیفیت باید در تمام مراحل طراحی (از جمله معماری)، پیاده‌سازی، و استقرار مد نظر قرار گیرد. کیفیت هم جنبه‌های مربوط به معماری و هم جنبه‌های نامربوط به معماری دارد. مثلاً دسترس پذیری با انتخاب از بین عناصر در برابر پشتیبانی از عمل بازگشت قابل حصول است و کارایی مقدار ارتباط بین مولفه‌ها در برابر اجرای الگوریتم‌های سیستم را تبیین می نماید. اصولا صفات کیفیتی مستقل نیستند و نباید بصورت مجزا به آنها دست یافت. در مراجع مختلف درخصوص صفات کیفیتی رده بندی های مختلفی ارائه شده است که در این بخش به مهم ترین آنها اشاره می نمائیم [۱۰و ۹و ۸]:

- صفات کیفیتی استاتیک که انعکاس ساختار یک سیستم و سازمان که مستقیماً با معماری، طراحی و کد منبع مرتبط است. این موارد برای کاربر نهایی نامرئی هستند اما بر هزینه توسعه و نگهداری تاثیر می گذارند. به عنوان مثال ماژولار بودن، قابلیت تست، قابلیت نگهداری و ...
- صفات کیفیتی پویا که این ویژگی ها بازتاب رفتار سیستم در هنگام اجرای آن است و مستقیما با معماری، طراحی، کد منبع، پیکربندی، پارامترهای استقرار، محیط و بستر سیستم ارتباط دارند. این موارد برای کاربر نهایی قابل مشاهده هستند و در زمان اجرا وجود دارند. به عنوان مثال توان عملیاتی، قدرت، مقیاس پذیری.
- صفات کیفیتی سیستمی مانند دسترس‌پذیری، تغییرپذیری، کارایی، امنیت، آزمون‌پذیری و قابلیت استفاده صفات کیفیتی عملکردی و کاری مانند مدت زمان بازاریابی صفات کیفیتی وابسته به معماری مانند یکپارچگی مفهومی.
- صفات کیفیتی قابل مشاهده از طریق اجرا مانند کارایی و امنیت.
- صفات کیفیتی غیرقابل مشاهده از طریق اجرا مانند تغییرپذیری و آزمون‌پذیری.
- صفات کیفیتی مبتنی بر دسترس‌پذیری مانند قابلیت اطمینان.
- صفات کیفیتی مبتنی بر تغییرپذیری مانند قابلیت حمل، قابلیت استفاده مجدد و مقیاس‌پذیری.
- صفات کیفیتی مبتنی بر قابلیت استفاده مانند قابلیت خود انطباقی و قابلیت انطباق کاربر.
- صفات کیفیتی با همبستگی مثبت مانند تغییرپذیری و قابلیت ساخت.
- صفات کیفیتی با همبستگی منفی مانند قابلیت اطمینان و امنیت.

آنچه اهمیت دارد این است که در تمام رده های ذکر شده، بهبود صفات کیفیتی منجر به عملکرد موثرتر سیستم نرم افزاری و افزایش هم افزائی در جهت دستیابی به اهداف سیستم هدف و درواقع تضمین کیفیت نرم افزار[1] می گردد. در بخش بعدی به ابعاد مساله تضمین کیفیت نرم افزار می پردازیم.

اندازه گیری کیفیت سیستم های نرم افزاری

موفقیت یک محصول نرم افزاری بطور خاص تحت عوامل متعددی قراردارد. باتوجه به اینکه سازمان ها و موسسات مختلف ازمحصولات نرم افزاری استفاده می کنند، لزوم دراختیار داشتن یک نرم افزار با کیفیت و مطلوب باتوجه به اهداف و نیازهای سازمان باعث میشود که اندازه گیری کیفیت محصولات و سیستم های نرم افزاری به یک مسئله مهم برای اکثرسازمان ها و موسسات تبدیل شود تا ازداشتن یک نرم افزار مطلوب مطمئن باشند. در این راستا، لازم است که برای مطالعه دقیق و اصولی دربحث کیفیت از مدل های کیفی استاندارد برای بررسی خصوصیات مرتبط استفاده نمود. در حال حاضر اغلب نرم‌افزارها از معماری چند لایه ای و پیشرفته استفاده می کنند، بنابراین تجزیه و تحلیل و اندازه گیری کیفیت نرم‌افزار باید به صورت جامع و مداوم مدیریت شود و از هدف نهایی یا استفاده نهایی نرم‌افزار جدا شوند. امروزه در سیستم های نرم افزاری اندازه گیری کیفیت حداقل با دو هدف اساسی صورت می گیرد:

[1] Software Quality Assurance (SQA)

مدیریت ریسک و مدیریت هزینه که در ادامه مطالب مقاله به توصیف آنها می پردازیم.

مدیریت ریسک

از گذشته تا کنون خرابی نرم‌افزار مشکلاتی بیش از ناراحتی و نگرانی ایجاد کرده است. مشکلات نرم‌افزاری گاها فجایای انسانی نیز ایجاد کرده است. این مشکلات میتواند از طراحی رابط کاربر ضعیف تا خطاهای اساسی نرم‌افزار در کد نویسی متغیر باشد. ریسک یک رویداد یا شرایط غیر قطعی است که اگر اتفاق بیفتد روی اهداف پروژه تأثیر می گذارد و این اهداف می تواند در حوضه رضایتمندی هزینه و کیفیت باشند. مدیریت ریسک یک پروسه نیازمند نظم و ترتیب بسیار گسترده و طولانی است. هر فردی که با مدیریت پروژه، فعالیت‌های میانی پروژه‌ها و توسعه نرم‌افزار سر و کار دارد، باید با مباحث مدیریت ریسک و انواع روش‌های مناسب برای کاهش ریسک در پروژه‌های مهندسی نرم‌افزار آشنا باشد. آنچه مهم است عدم وجود یک روش و پیشنهاد منحصربه‌فرد برای مدیریت ریسک در پروژه‌های نرم‌افزاری مختلف است. مدیران پروژه باید بر اساس مختصات پروژه نرم‌افزاری، یک یا چند روش مدیریت ریسک را انتخاب کرده و به کار گیرند. اغلب افراد با تجربه‌اندوزی و داشتن مهارت و ممارست در مدیریت ریسک می‌توانند بهترین استراتژی را در مدیریت ریسک انتخاب و پیاده‌سازی کنند [۱۲و۱۱].

معمار نرم افزار نیز می تواند با بکارگیری یک برنامه مدون اجتناب از ریسک[۲] و انجام ندادن فعالیت هایی که باعث ریسک می شود، از چالش های آتی جلوگیری نماید. همچنین مدیر پروژه ممکن است اهداف پروژه را از تأثیر ریسک محفوظ دارد، یا اهدافی که در معرض خطر قرار دارند را مشخص نماید، مانند توسعه زمانبندی، تغییر استراتژی یا کاهش حوزه است. بعضی از ریسک ها در همان ابتدای پروژه نمایش داده می شود و می توانند از طریق دسته بندی نیازها، بدست آوردن اطلاعات و بهبود روابط خنثی شوند.

پس از شناسایی و طبقه‌بندی ریسک‌های احتمالی برای یک پروژه نرم‌افزاری، مدیر پروژه باید یک طرح و برنامه برای مدیریت ریسک‌ها ترسیم کند. مدیر پروژه (به‌عنوان یک بخش از طرحی بزرگ‌تر و جامع‌تر، در طرح و برنامه) باید برای هر ریسک یک پاسخ مشخص پیش‌بینی‌شده داشته باشد تا تاثیرات آن را کاهش دهد. اساسا مدیریت ریسک شامل مراحل زیر می‌باشد:

- شناسایی ریسک‌ها و عوامل آن‌ها
- طبقه‌بندی و اولویت‌دهی تمام ریسک‌ها
- طرح‌ریزی برای کاهش تاثیرات ریسک‌ها
- مانیتور عوامل ریسک‌ها هنگام انجام پروژه
- پیاده‌سازی اقدامات کاهش ریسک در صورت بروز
- ارتباطات وضعیت ریسک در سراسر پروژه

در این راستا، برخی از تمرین‌ها و ممارست‌هایی که می‌توانند قدرت مدیریت ریسک را افزایش دهند به شرح زیر هستند:

- همیشه در هر وضعیتی از پروژه به مدیریت ریسک فکر کنید، درغیراینصورت تیم پروژه نرم افزاری از یک بحران وارد بحران دیگر خواهد شد.
- از چک‌لیست‌ها استفاده و آن‌ها را با پروژه‌های مشابه دیگر مقایسه کنید.
- حتما ریسک‌ها را بر اساس سطح تهدیدات آن‌ها اولویت‌بندی کنید.
- فهرست‌های ۱۰ ریسک و ۲۰ ریسک با اولویت بالا برای پروژه تهیه کنید. می‌توانید این فهرست‌ها را برای پروژه‌های متعدد استفاده و تغییر و بهبود دهید.
- با برگزاری جلسات مختلفی با تیم‌ها و افراد درگیر با پروژه، به‌خصوص تیم‌های بازاریابی، خدمات پس از فروش و ارتباط با مشتری، سعی کنید با ریسک‌های جدیدی آشنا شوید.

در صورت امکان، ریسک‌های بزرگ و بسیار موثر در شکست یا موفقیت یک پروژه را به تعدادی ریسک کوچک‌تر بشکنید. معمولا ریسک‌های کوچک‌تر راحت‌تر تشخیص داده‌شده و قابل کنترل هستند. مدیران و سهام‌داران یک پروژه را مدام تشویق کنید درباره ریسک‌ها و مشکلات یک پروژه هم‌فکری کرده و مواردی را که به ذهن‌شان می‌رسد، گزارش دهند. همیشه صحبت کردن درباره یک پروژه و خطرات بالقوه آن می‌تواند ریسک‌های پهنان را آشکار کند [۱۳].

لازم به ذکراست، استراتژی مدیریت ریسک نرم افزار میتواند در برنامه طراحی معماری و سیستم پروژه قرار گیرد یا بصورت جداگانه در طرحی با نام طرح RMMM[۳] مستند گردد. طرح RMMM تمامی کارهای انجام گرفته تحت عنوان تحلیل ریسک را مستند کرده و توسط مدیر پروژه بعنوان بخشی از طرح پروژه کلی مورد استفاده قرار میگیرد.

مدیریت هزینه

مدیران و خبرگان دنیای نرم افزار بر این عقیده اند که کیفیت بالای محصول نرم افزاری به صرفه جویی و مدیریت در هزینه ها و ارتقاء همیشگی سطح نرم افزار منتج می شود. این در حالیست که تمامی توسعه دهندگان نرم افزاری توافق دارند که دستیابی به نرم افزار های با کیفیت، بالاترین هدف در ایجاد وساخت سیستم های نرم افزاری است. همانند زمینه های دیگر مهندسی،

[2] Risk Avoidance Plan

[3] Risk Mitigation, monitoring & Management Plan

برنامه ای با کیفیت ساختاری مناسب هزینه نگهداری، یادگیری و تغییر بسیار کمتری دارد. آمار نشان می دهد که کیفیت ساختاری ضعیف در برنامه های تجاری مانند برنامه ریزی منابع سازمانی (ERP)، مدیریت ارتباط با مشتری (CRM) و یا سیستم های مالی با پردازش های بزرگ باعث افزایش هزینه ها می شود و از طرفی باعث ایجاد دوباره کاری (حتی تا ۴۵ درصد از زمان توسعه در برخی سازمانها) می شود. علاوه بر این، کیفیت ساختاری ضعیف به دلیل ارائه ی داده های غلط، توقف برنامه ها و مشکلات امنیتی و پردازشی باعث ایجاد اختلال در عملکرد سازمان ها می شود [۱۵و ۱۴].

اساسا هزینه کیفیت شامل تمام هزینه های صرف شده برای رسیدن به کیفیت و یا انجام فعالیت های مربوط به آن است. هزینه‌های کیفیت می توانند به هزینه های مربوط به پیشگیری، شکست و ارزیابی تقسیم شوند. هزینه های پیشگیری شامل آموزش، وسائل و تجهیزات آزمایش و برنامه ریزی کیفیت است. هزینه شکست مربوط به دوباره کاری، تعمیر و تحلیل وضعیت شکست خواهد شد و هزینه ارزیابی مشتمل بر بازبینی فرآیندها و آزمایش خواهد بود. لازم به ذکر است، بین اندازه گیری و بهبود کیفیت نرم‌افزار در یک سیستم جاسازی شده (با تأکید بر مدیریت ریسک) و کیفیت نرم‌افزار در نرم‌افزارهای تجاری (با تأکید بر هزینه و مدیریت پایداری) تفاوت وجود دارد. سیستم های جاسازی شده در حال حاضر اغلب شامل یک رابط کاربری هستند و طراحان آنها به همان اندازه که همتایان خودروی برنامه های تجاری متمرکز هستند، نگران قابلیت استفاده و بهره وری کاربر هستند. گروه دوم به دنبال سیستم ERP یا CRM به عنوان یک سیستم شرکتی هستند که زمان و عملکرد آن برای رفاه شرکت مهم است. این همگرایی بیشتر در پردازش های موبایل قابل مشاهده است: کاربری که به یک برنامه ERP در تلفن هوشمند خود دسترسی پیدا می کند، از بین لایه های نرم‌افزار به کیفیت وابستگی دارد [۱۷و۱۶].

تضمین کیفیت نرم‌افزار

پس از مباحث مطرح در چگونگی ایجاد محصول نرم‌افزاری و مباحث مربوطه؛ یکی از چالش‌های موجود در این حوزه بررسی مفهوم چیستی کیفیت نرم‌افزار و متعاقب آن معیارهای سنجش و روش‌های تضمین کیفیت آن است. اساسا تضمین کیفیت نرم‌افزار عبارتست از نظارت بر روند مهندسی نرم‌افزار و روش‌هایی که برای اطمینان یافتن از کیفیت آن مورد استفاده قرار می‌گیرند. در این حوزه، روش‌هایی که بدین منظور ایجاد شده‌اند بسیار زیاد و متنوع هستند که هریک انطباق با چند مورد از استانداردهای رایج را بررسی و تضمین می‌کنند. روش‌های تضمین کیفیت نرم‌افزار، کل فرایند توسعه نرم‌افزار را دربرمی‌گیرند. از جمله مراحل توسعه نرم‌افزار می‌توان به تعریف نیازمندی‌ها، طراحی معماری نرم‌افزار، کدنویسی، بازبینی کد، مدیریت پیکربندی سیستم نرم‌افزاری، تست، مدیریت پخش، یکپارچه سازی محصول اشاره کرد. تضمین کیفیت نرم‌افزار به اهداف، الزامات، توانایی‌ها، فعالیت‌ها، اندازه‌گیری‌ها و ارزیابی‌ها سامان می‌بخشد. در واقع فرایند تضمین کیفیت یک فرایند حمایتی است که باید به وسیله طرح‌های از پیش تعیین شده ابعاد مساله کیفیت را در تک تک محصولات، فعالیت‌ها و روال ها به‌طور مستقل تضمین کند. تضمین کیفیت نرم‌افزار شامل محدوده گسترده ای از فعالیت‌ها می‌شود.

استانداردها

در این حوزه سازمان هائی همچون IEEE و ISO ، لیست گسترده ای از استانداردهای محصولات نرم‌افزاری و مستندات مربوط به آن معرفی کرده‌اند. به‌طور کلی استانداردها یا توسط سازمان تولیدکننده نرم‌افزار به‌طور داوطلبانه بر روی محصول اعمال شده یا اعمال آن‌ها از جانب خریدار و سایر ذینفعان محصول نرم‌افزاری، به شرکت تولیدکننده تحمیل می‌شود. وظیفه تیم SQAآنست که از انطباق محصول با استانداردهایی که برای محصول به تصویب رسیده است، اطمینان حاصل کنند [۱۸].

نظارت بر کیفیت

نظارت بر کیفیت یک سیستم نرم افزاری از ابعاد مختلف و با شیوه‌های متفاوتی می‌تواند صورت بگیرد. نظارت بر کیفیت سیستم‌های نرم‌افزاری را از ابعاد مختلف کارکردی و غیرکارکردی (خصوصاً کارایی و امنیت) و با شیوه‌های جعبه سیاه[4] و جعبه سفید[5] انجام می‌پذیرد. در شیوه جعبه سیاه، بررسی کیفیت سیستم از طریق اجرای تست‌های مختلف و بدون دسترسی به مستندات مشروح و کد منبع صوت می‌گیرد. در مقابل، در شیوه جعبه سفید، بررسی کیفیت سیستم در شرایطی که امکان دسترسی به مستندات مشروح تحلیل، طراحی و پیاده‌سازی سیستم وجود دارد، صورت گرفته و خطاها و نقاط ضعف سیستم به صورت دقیقتر شناسایی می‌گردد [۲۰و۱۹].

بازبینی و ممیزی

بازبینی فنی یکی از فعالیت‌های کنترل کیفیت صورت گرفته توسط مهندسین کامپیوتر است. هدف این بازبینی‌ها رفع ایرادات موجود است. ممیزی نیز نوعی بازبینی محسوب می‌شود. با این تفاوت که توسط کارکنان SAQ با هدف اطمینان حاصل کردن

[4] Black-Box

[5] White-Box

از اینکه محصول از دستورالعمل‌های کیفی پیش فرض پیروی کرده است، انجام می‌شود.

تست و آزمون

آزمون نرم‌افزار یکی از فعالیت‌های کنترل کیفیت است که هدف اولیه آن پیدا کردن خطاهای عملکردی است. وظیفه SQA آنست که از درست برنامه‌ریزی شدن فرایند تست و بهینگی اجرای روال ها و عملیات اطمینان حاصل کند [۳].

مدیریت تغییر

تغییر یکی از مخرب‌ترین جنبه‌های یک پروژه نرم‌افزاری است. درصورتی که تغییرات به درستی مدیریت نشوند می‌توانند سبب گیجی و سردرگمی شده و این سردرگمی اغلب موارد به کم کیفیت شدن محصول نرم افزاری خواهد انجامید. در این زمینه SQAوظیفه دارد از برقراری و اعمال روش‌های مناسب مدیریت تغییر اطمینان حاصل کند.

مدیریت امنیت

با افزایش جرائم اینترنتی و قوانین جدید دولت‌ها در مورد حریم خصوصی، هر سازمان نرم‌افزاری باید سیاست‌هایی را سازمان نهاده تا از داده‌های تمامی سطوح نرم‌افزاری کاربران محافظت کند. به‌طور مثال برای محافظت برنامه‌های وب از دیوار آتش استفاده کند یا مطمئن شود اجزاء نرم‌افزار مورد دستکاری داخلی قرار نگرفته باشد. وظیفه SQA در این حوزه آنست که از بکارگیری فرایندها و تکنولوژی‌های مناسب به منظور حفظ امنیت کاربران اطمینان حاصل کند[۱۹].

مدیریت ایمنی

نرم‌افزار همیشه یک مؤلفه محوری سیستم‌هایی که برای کاربری انسانی استاندارد شده‌اند می‌باشد، از همین رو تأثیر نقص‌های پنهان آن می‌تواند فاجعه بار باشد. مسئولیت SQA آنست که ضمن ارزیابی تأثیر خرابی نرم‌افزار، اقدام‌های لازم برای کاهش ریسک بروز این خرابی را انجام دهد. اگرچه تحلیل و مقابله با ریسک از جمله دغدغه‌های مهندسین نرم‌افزار است، سازمان SQA نیز بر اتخاذ صحیح اقدامات کاهش ریسک و برقراری طرح‌های مقتضی مرتبط با ریسک نظارت می‌کند. علاوه بر این دغدغه‌ها و فعالیت‌ها یکی از نقش‌هایSQA ، اطمینان حاصل کردن از انجام با کیفیت فعالیت‌های پشتیبانی نرم‌افزار (مثل نگهداری، مستندسازی، راهنمای کاربری) است. در بخش قبل درخصوص ابعاد مختلف مدیریت ریسک به تفضیل صحبت شد.

نتیجه گیری

دستیابی به نرم افزار با کیفیت بالا حتی برای بی تجربه ترین توسعه دهندگان نرم افزار یک هدف مهم است. درجه تطابق یک سیستم با نیازمندی های مشخص شده و نیازهای کاربران یا انتظارت آن ها، یکی از تعاریف رسمی کیفیت در نرم افزار است. جهت رسیدن به کیفیت بالا، به معیار هایی برای اندازه گیری و سنجش نیازمند هستیم. بنابراین جامع بودن این معیار ها که تمامی جزئیات مربوط به فرآیند را پوشش می دهد می تواند منبعی مناسبی جهت رسیدن به این هدف باشد. دیدگاه های جدید در مورد کیفیت نرم افزار با هدف بیشینه کردن ارزش ها در نرم افزارها است و ارزش ها خواسته های مشتری در نرم افزار و انطباق مشخصه های کلیدی با ترجیهات مشتری بر اساس اولویتها است. بنابراین گسترش عملکرد کیفیت برای ارزیابی ارزش های مشتری بکار برده می شود و در واقع تعادلی بین استراتژی های توسعه و هدایت بوجود می آورد. در این مقاله درخصوص معیارهای کیفی نرم‌افزار، استانداردها و مدل های رایج و نحوه اندازه گیری و دستیابی به این معیارها در مراحل طراحی معماری و سیستم نرم افزاری به تفضیل صحبت نمودیم. درک صحیح ابعاد این مساله مهم می تواند در جهت توسعه و بکارگیری نرم افزارهای موثر و کارا با سطح کیفی مطلوب راهگشا باشد.

منابع

[1]. X. Han, R. Li, W. Li, G. Ding, and S. Qin, "User requirements dynamic elicitation of complex products from social network service". International Conference on Automation and Computing (ICAC), pp. 1–6, 2019.

[2]. L. Zhao, A. Zhao. "Sentiment analysis based requirement evolution prediction." Future Internet, IEEE, vol. 11, no. 2, p. 52, 2019.

[3] S. Burnay, I. Faulkner, et al. "A Discussion of the Impact of Judgmental Heuristics on Elicitation during Requirements Engineering", *Proceedings 13th IEEE International Conference on Research Challenges in Information Science RCIS 2019 Brussels*, 2019.

[4] Z.S. Shaukat, R. Naseem, et al. "A dataset for software requirements risk prediction". In Proceedings of the 2018 IEEEInternational Conference on Computational Science and Engineering (CSE), Bucharest, Romania, 29–31 October pp. 112–1,2018.

[5] T. Ambreen, N. Ikram, M. Usman, M. Niazi, "Empirical research in requirements engineering: trends and opportunities", Requirements Engineering 23(1): 63-95, 2018.

[6]. G. Liebel, M. Tichy, E. Knauss, O. Ljungkrantz, G. Stieglbauer, " Organisation and communication problems in automotive requirements engineering", Requirements Engineering 23(1): 145-167, 2018.

[7] S. Wiesnera, S. Nilssonb, K. Thobena, "Integrating requirements engineering for different domains in system development– lessons learnt from industrial SME cases", Elsavier 2017.

[8]. H. Scherer, A. Albers, N. Bursac, "Model Based Requirements Engineering for the Development of Modular Kits", Elsavier, 2017.

[9] S. Maalema, N. Zarourb, "Challenge of validation in requirements engineering", Elsavier, 2016.

[10] A. Safwat and M.B.Senousy, "Addressing Challenges of Ultra Large Scale System on Requirements Engineering", Elsavier, 2015.

[11] S. Wiesner, M. Peruzzini, J. B. Hauge, K.D Thoben, "Requirements Engineering". In: Stjepandić J, Wognum N, Verhagen WJ, editors. Concurrent engineering in the 21st century: Foundations, developments and challenges. Springer, p. 103–132, 2015.

[12] W. Chang, W. Yan, "Customer Requirements Elicitation and Management for Product Conceptualization". In: Stjepandić J, Rock G, Concurrent Engineering Approaches for Sustainable Product Development in a Multi-Disciplinary Environment. London: p. 957–968, Springer, 2013.

[13] A. Przybyłek, "A Business-Oriented Approach to Requirements Elicitation", in Proceedings of the ۹th International Conference on Evaluation of Novel Software Approaches to Software Engineering(ENASE'14), pp. 152-163, 2014.

[14] S. Ling Lim, C. Ncube, "Social Networks and Crowdsourcing for Stakeholder Analysis in System of Systems Projects", Proc. of the 2013, 8th International Conference on System of Systems Engineering, Maui, Hawaii, USA - June 2-6, 2013.

[15] M. Sourour, "Metodology de validation des exigences collaborative dans les organisations distribuées ", International Workshop on Requirements Engineering Constantine- Algeria, 2013.

[16] Y. Constantinidis, "Expression des Besoins pour le Système d'Information_Guide d'élaboration du Cahier des Charges", Edition Eyrolles, 2012.

[17] A. Khosravi, N. Modiri, "Requirement Engineering in Service-Oriented Architecture", International Conference on Networks and Information, 2012.

[18] M. Attarha, N. Modiri, "Focusing on the importance and the role of requirement engineering". In: The 4th International Conference on Interaction Sciences, 16-18 Aug, 2011.

[19] M. Berkovich, JM. Leimeister, H. Krcmar, "Requirements Engineering for Product Service Systems", Wirtschaftsinf:357-370, 53(6), 2011.

[20] C. Monsalve, A. April, A. Abran, "Representing Unique Stakeholder Perspectives in BPM Notations". In: 8th ACIS International Conference on Software Engineering Research, Management and Applications, Montreal, Canada, 2010.