
بسم الله الرحمن الرحیم

واحد علوم و تحقیقات

خصوصیات کیفی نرم افزار

مهندسی نرم افزار پیشرفته

تنظیم و ویرایش : محسن زاده

فهرست مطالب

۱- مقدمه و تاریخچه:

امروزه سیستم‌های کامپیوتری در بسیاری از کاربردهای حساس و بحرانی انسان نقش دارند، به طوری‌که یک خطای کوچک می‌تواند منجر به بروز مشکلات جدی و خطرناکی شود. این خطا ها می‌تواند به صورت یک خطا در طراحی نادرست واسط کاربر تا خطای موجود در کد برنامه باشد .

کاربردهای بحرانی خصوصیات زیر را دارا می‌باشند:

- چرخه عمر طولانی دارند و نیازمند بهبود تکاملی می‌باشند.
- کاربردها نیاز به اجرای پیوسته هستند.
- کاربردها نیاز به تعامل با دستگاه‌های سخت افزاری هستند.
- عملکرد کاربرد بستگی به ویژگی‌های کیفی مانند دقت بالا، قابلیت اطمینان، ایمنی، قابلیت همکاری با بخش‌های دیگر دارد.

سیستم‌ها دارای نیازمندی‌های گوناگون می‌باشند که در واقع توسعه‌دهندگان مسئول تشخیص و تعیین این نیازمندی‌های و همچنین ایجاد سیستمی برای محقق‌سازی این نیازمندی‌ها می‌باشد.

این نیازمندی‌ها در حالت کلی به دو دسته نیازمندی‌های کارکردی[۱] و نیازمندی‌های غیر کارکردی[۲] تقسیم می‌شوند. نیازمندی‌های کارکردی توانایی سیستم برای انجام کاری است که برای آن بوجود آمده است . باید بسیاری از عناصر سیستم با هم هماهنگ عمل کنند تا یک کار تکمیل شود .

نیازمندی‌های غیر کارکردی همان مشخصه‌های کیفیتی [۳] می‌باشند و عموماً به‌صورت ضمنی بیان می‌شوند و هر آنچه که غیر از نیازمندی‌های کارکردی سیستم باشد، در این دسته قرار می‌گیرند مانند کارایی، امنیت، قابلیت استفاده مجدد، قابلیت اطمینان و . . . .

هدف این جزوه بررسی اجمالی چند مدل موجود برای نیازمندی‌های کیفی و بررسی تعدادی از خصوصیات کیفی می‌باشد.

---

۱ Functional
۲ Non Functional
۳ Quality Attribute

اما به دلایل زیر مسئله خصوصیات کیفی در نرم‌افزار مسئله‌ای پیچیده‌ای است:

- تنش و بحرانی که بین نیازمندی‌پهای کیفی مشتریان (مانند کارایی، قابلیت اطمینان و . . .) و نیازمندی های کیفی تولید کنندگان (قابلیت نگهداری و استفاده مجدد) وجود دارد.
- برای بیشتر نیازمندی های کیفی روشی روشن و بدون ابهام جهت تعریف آنها وجود ندارد.
- خصوصیات کیفی نرم افزار معمولا متناقض هستند. (مثلاً کارایی و امنیت)

دلایل اهمیت خصوصیات کیفی به قرار زیر است:

- خصوصیات سیستم را توصیف می‌کند.
- شرایط کاری سیستم را بیان می‌کند.
- عامل و محرک طراحی معماری می‌باشد.
- معماری اولین قدم تولید نرم‌افزار بوده که ویژگی‌های کیفیتی در آن قابل ردیابی است. صفات کیفیتی در تمام مراحل طراحی، پیاده سازی و انتقال مطرح می‌باشند و در نتیجه اگر توسط معماری پشتیبانی شوند، راحت‌تر قابل ردیابی خواهند بود.

۲- مشخصات یک محصول نرم‌افزاری

انجام خصوصیات کیفی باید در مراحل طراحی، پیاده‌سازی و توسعه مورد توجه قرار گیرد. خصوصیات کیفی نه کاملاً به مرحله طراحی و نه کاملاً به مرحله پیاده‌سازی وابسته است بلکه به تمام این مراحل بستگی دارد. یک مجموعه سه‌تایی از خصوصیات یک محصول نرم افزاری از دیدگاه مشتری عبارت است از:

- کیفیت[۴]
- هزینه[۵]
- زمان‌بندی[۶]

[۴] Quality
[۵] Cost

هزینه و زمان قابل پیش‌بینی می‌باشد و با یک فرایند کامل سازماندهی شده قابل کنترل است ولی کیفیت این‌گونه نمی‌باشد. طبق استاندارد IEEE"کیفیت نرم افزار درجه‌ای است که نرم‌افزار ترکیبی از خصوصیات مطلوب را دارا می‌باشد." همچنین در استاندارد ISO 8402" کیفیت، توانایی برآوردن نیازهای تعیین شده و تلویحی می‌باشد". کیفیت پائین به جهت امکان نیاز به پیکربندی،کدنویسی مجدد و حتی طراحی دوباره، بر روی هزینه و زمان‌بندی تأثیر می‌گذارد. کشف سریع‌تر مشکلات کیفی در نرم افزار مقرون به صرفه‌تر می‌باشد زیرا با افزایش زمان نقص در نرم افزار با توجه به بزرگتر شدن پروژه هزینه لازم برای حل مشکل نیز بیشتر می‌شود.

دست یافتن به خصوصیات کیفی تنها وابسته به شیوه کد برنامه (مانند انتخاب زبان، الگوریتم و ساختمان داده و. . .) نیست بلکه وابسته به معماری نرم افزار نیز می‌باشد.

۲-۱ مدیریت کیفیت نرم افزار[۷]:

مدیریت کیفیت نرم افزار به مسئله مهم ذیل توجه دارد:

- تضمین و اطمینان از برآوردن سطح کیفی مورد نظر در محصول نرم افزاری.
- تعریف استانداردها و رویه‌های مناسب کیفی.
- تلاش جهت توسعه یک فرهنگ کیفی[۸] مناسب.

بحث مدیریت کیفیت نرم‌افزار باید از بحث مدیریت پروژه کاملاً جدا شود. در زمینه مدیریت کیفیت فعالیت‌هایی انجام می شود که عبارتند از:

- **تضمین کیفیت**[۹]: رویه‌های کیفی و استانداردهایی برای کیفیت تعریف می‌کند. انجام فرایند بررسی، تصحیح و کنترل به طوری که تولیدکننده مطمئن باشد که تمام مراحل فرایند تولید به درستی انجام شده و خروجی‌ها همانگونه که انتظار داشته است، می‌باشند.

---

[۶] Schedule
[۷] Software Quality Management
[۸] quality culture

- **برنامه‌ریزی کیفی**[10]: رویه‌های قابل اجرا و استانداردهایی را برای پروژه‌های بخصوص انتخاب می‌کند و در همان راستای مورد نیاز آنها را اصلاح و ویرایش می‌کند. در واقع در برنامه‌ریزی کیفی خصوصیات کیفی مورد نظر محصول و چگونگی ارزیابی آنها مورد بررسی قرار می‌گیرد و خصوصیات مهم نیز تعریف می شود.
- **کنترل کیفی**[11]: اطمینان از اینکه استانداردها و رویه‌های کیفی توسط تیم توسعه‌دهنده مورد استفاده قرار می‌گیرد. دو رهیافت مورد استفاده در این مرحله عبارتند از:
  - بازبینی و مرور کیفیت که به صورت دستی انجام می شود. انواع بازبینی‌ها عبارتند از:
    - بازبینی طراحی برنامه[12] : جهت کشف خطاهای موجود در طراحی ،کد یا نیازمندی‌ها انجام می شود. در این مرحله تعدادی از خطاهای احتمالی کشف می شود.
    - بازبینی پیشرفت کار[13]: فراهم سازی اطلاعاتی برای مدیریت درباره کل فرایند پروژه می‌باشد. بازبینی شامل بازبینی محصول و فرایند است. این مرحله با هزینه، زمان‌بندی و طراحی درگیر می‌باشد.
    - بازبینی کیفی[14]: انجام یک تحلیل تکنیکی از مؤلفه‌های محصول یا مستندات جهت یافتن عدم تطابق‌هایی که بین خصوصیات و طراحی مؤلفه،کد یا مستندات و اطمینان از اینکه استانداردهای کیفی رعایت شده است. در طول فرایند بازبینی و مرور کیفی، رویدادها را می‌توان به چند دسته و به صورت زیر دسته‌بندی کرد:
      - ❖ هیچ عملیاتی انجام نمی شود وتغییری بر روی نرم افزار و مستندات نیاز نمی‌باشد.
      - ❖ نیاز به ترمیم دارد: در این حالت طراح و برنامه نویس باید خطاهای شناسایی شده را تصحیح نماید.

---

[9] Quality Assurance
[10] Quality Planning
[11] Quality Control
[12] Design Inspection
[13] Progress Inspection
[14] Quality Inspection

❖ در کل طراحی باید تجدید نظر انجام شود: در این حالت مشکل شناسایی شده در طول مرحله بازبینی بر بخش‌های مختلف تأثیر می‌گذارد و باید به دنبال راه‌هایی مقرون به صرفه‌تر برای حل مشکل طراحی مجدد بود.

- ارزیابی خودکار نرم افزار (به صورت خودکار انجام می شود): با استخراج یک مقدار عددی برای خصوصیات محصول یا فرایند نرم افزاری سروکار دارد. اندازه‌گیری کیفیت در طول چرخه حیات محصول، به منظور مدیریت کیفیت، اندازه گیری و ارزیابی کیفیت محصول و فرآیند انجام می‌شود. اندازه‌گیری کیفیت احتیاج به جمع‌آوری و تجزیه و تحلیل اطلاعات دارد که معمولا به شکل اندازه‌ها (Measurements) و معیارها (Metrics) نمایان می‌شوند. ارزیابی کیفیت معمولاً زمانی که یک رویداد مهم رخ می‌دهد، مثلاً در انتهای یک مرحله از تولید یا زمان انتشار محصول نهایی، انجام می‌شود.

۲-۲ مدل‌های کیفیتی در نرم افزار

از آنجائی‌که کیفیت نرم افزار نمودهای متعددی در زندگی روزانه ما دارد، رویکردهای متعددی در ارتباط با مدل‌های کیفی منتشر شده است. مدل‌های کیفیتی دسته‌بندی از صفات کیفیتی و ارتباطات میان آنها می‌باشد که برای مشخص نمودن و ارزیابی نیازمندی‌های غیر وظیفه‌ای بکار می‌روند. در ادامه چند مدل کیفی مختصراً ارائه شده است .

۲-۲-۱ مدل McCall

در دهه ۱۹۷۰ **McCall** چارچوبی کیفیتی برای کیفی محصولات ارائه نمود. این مدل برای نیروی هوایی آمریکا و با هدف از بین بردن فاصله بین کاربران و توسعه‌دهندگان[۱۵] ایجاد شد . این مدل ۳ دیدگاه را برای گروه‌بندی خصوصیات در فرآیند تولید نرم افزار بیان می‌کند که بر اساس چرخه حیات سیستم می‌باشد، این دیدگاه‌ها عبارتند از:

❖ عملکرد[۱۶](خصوصیات عملکردی پایه): فاکتورهایی کیفی می‌باشدکه بر میزان و اندازه براورده ساختن مشخصات و خصوصیات نرم افزار دلالت دارد. این خصوصیات عبارتند از:

---

[۱۵] Developer
[۱۶] Operation

- **Correctness**: صحت و درستی نرم افزار می باشد و اینکه عملکرد با خصوصیات ومشخصات تطبیق داشته باشد.
- **Reliability**: معیاری است که سیستم چقدر **fail** می کند.
- **Efficiency**: کاربرد و استفاده از منابع سیستم (شامل **Memory،Disk، CPU** و . . .)
- **Integrity**: حفاظت در مقابل انجام اعمال‌های غیر مجاز
- **Usability** : سادگی استفاده

❖ *تجدید نظر*[۱۷]: فاکتورهای کیفی را تعریف می‌کند که بر توانایی تغییر محصول نرم‌افزاری تأثیر می‌گذارند. این فاکتورها عبارتند از :

- **Maintainability**: توانایی جهت ساخت / تعمیر یک عیب / نقص می‌باشد.
- **Flexibility**: توانایی جهت اعمال تغییر درخواست شده در راستای **business** نرم افزار
- **Testability**: توانایی اعتبارسنجی نیازمندی‌های نرم‌افزاری و همچنین تست نرم افزار از جهت نداشتن خطا و برآوردن خصوصیاتش [۱۸]

❖ تحول[۱۹]: فاکتورهای کیفی می‌باشد که بر توانایی تغییر محصول نرم افزاری تأثیر می‌گذارد. این فاکتورها عبارتند از:

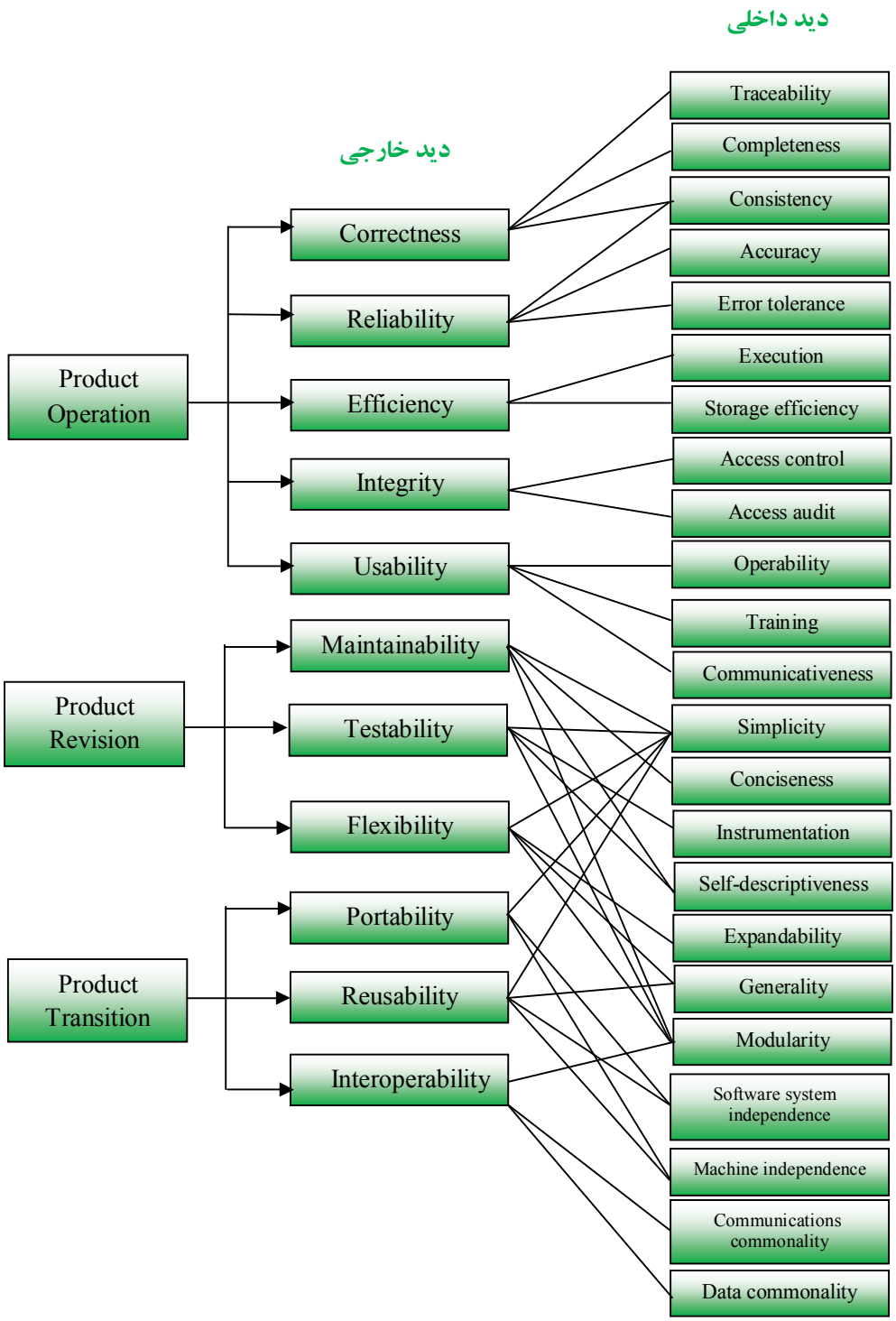
- **Reusability**: راحتی استفاده از مؤلفه‌های نرم افزاری موجود در زمینه‌ها و فضاهای مختلف.
- **Interoperability**: توسعه و راحتی مؤلفه‌های نرم افزاری جهت کار با یکدیگر (قابلیت تعامل).
- **Portability**: توانایی انتقال نرم افزار از یک محیط به محیط دیگر

این مدل از آن جهت مورد انتقاد قرار گرفت که خصوصیات کیفی در آن به صورت ذهنی مورد قضاوت قرار می‌گیرد و پاسخ‌ها کاملاً بستگی به نظر شخصی دارد. مدل **McCall** دارای ۱۱ فاکتور می‌باشد که دید خارجی سیستم یعنی همان چیزی که توسط کاربران دیده می شود را توصیف می‌کند. همچنین این مدل دارای ۲۳ فاکتور دیگر می‌باشد که دید داخلی یعنی همان دیدی را که توسط توسعه-دهندگان دیده می شود را توصیف می کند.

این چارچوب در شکل۱ نشان داده شده است. در این شکل خصوصیات به همراه سنجه‌ها و معیارهای اندازه‌گیری نشان داده شده است.

---

[۱۷] Revision
[۱۸] Meet Specification
[۱۹] Transition



شکل ۱: مدل **Mc Call**

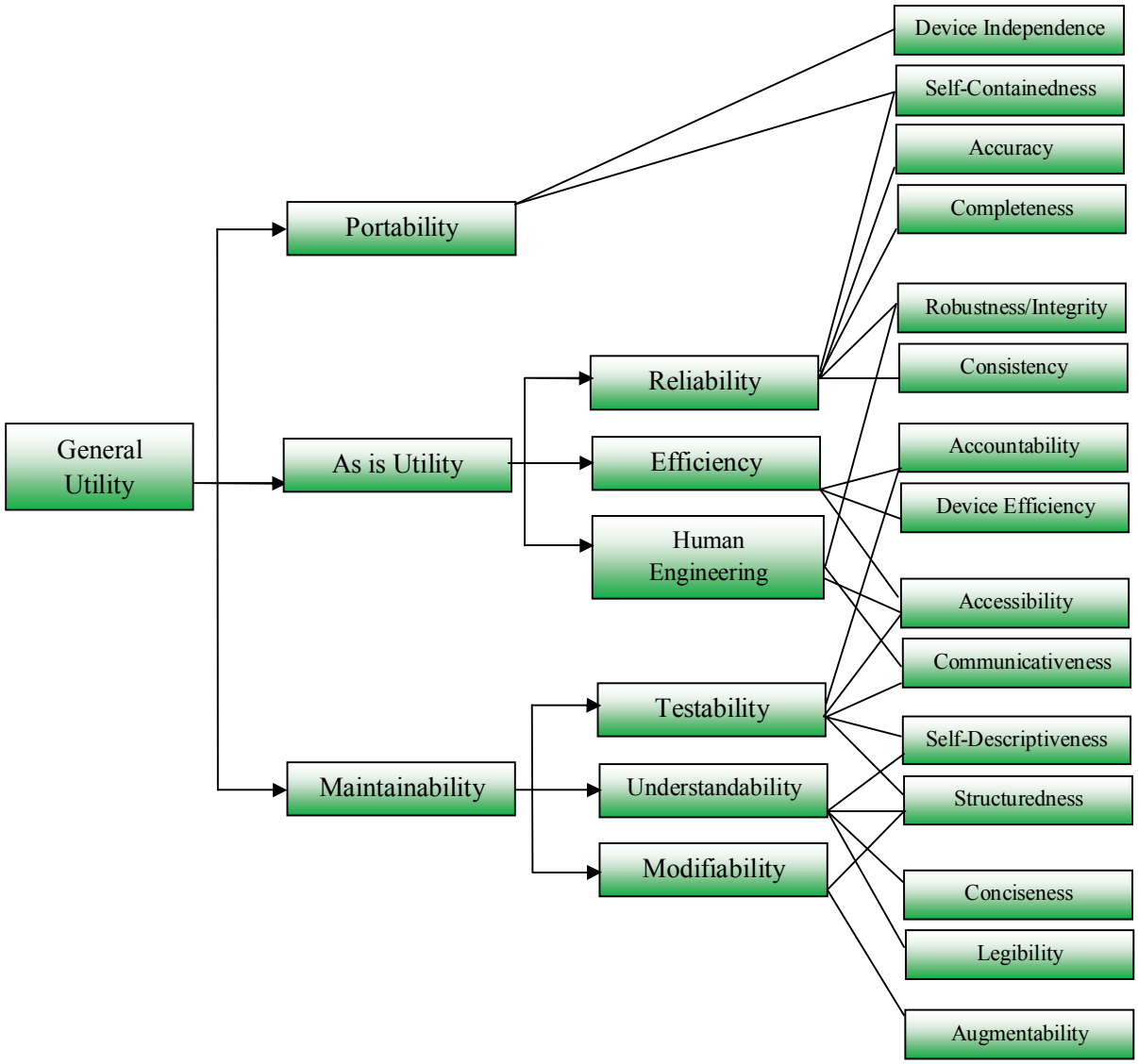
۲-۲-۲مدل Boehm

در سال ۱۹۷۸، **Boehm** مدل کیفیتی جدیدی را ارائه نمود. وی یک مدل سلسله مراتبی از خصوصیات کیفی را تعریف نمود و تلاش داشت تا کیفیت نرم افزار را بصورت مجموعه‌ای از خصوصیات و معیارها تعریف کند. در بالاترین سطح مدل، او سه نیازمندی اساسی نرم افزار را تعریف می‌کند که عبارتند از (شکل ۲)

- کاربر نهایی[۲۰]: (فضایی که نرم افزار موجود می‌تواند استفاده شود)
  - **Reliability**: نرم افزار وظایف خود را به صورت رضایت‌بخش انجام دهد.
  - **Efficiency**: آیا وظایف خود را بصورتی انجام می دهد که از به هدر رفتن منابع جلوگیری شود.
  - **Human Engineering**: زمانی که تغییری داده می شود،این تغییرات به آسانی قابل اعمال باشد.
- کاربران در مکان‌های مختلف (قابلیت حمل)[۲۱]: راحتی تغییر نرم افزار جهت تطبیق با محیط جدید
- کاربران در زمانهای مختلف (نگهداشت پذیری)[۲۲]: راحتی تعریف اینکه چه چیزی نیاز به تغییر دارد همراه با راحتی اصلاح و بازآزمایی (**retesting**)
  - **Testability**: راحتی ایجاد شرایط تصدیق .
  - **Understandability**: کد به راحتی قابل فهم باشد.
  - **Modifiability**: کدبه راحتی قابل تغییر و اصلاح باشد.

بزرگ‌ترین تفاوت بین **Boehm** و **McCall** این است که مدل **Boehm** بر مبنای رنج وسیعی از خصوصیات کیفی با یک دید اصولی روی "نگهداشت پذیری" می‌باشد. از طرفی **McCall** تمرکز دقیق‌تری روی اندازه‌گیری خصوصیت سطح بالای As is Utility دارد.

[۲۰] As is Utility
[۲۱] Portability
[۲۲] Maintainability



شکل ۲: مدل Boehm

۲-۲-۳ مدل ISO ۹۱۲۶

توسعه بیشتر مدل‌های کیفی منجر به ارائه یک مدل استاندارد توسط **ISO** شده است **ISO ۹۱۲۶** کیفیتی مدل استاندارد است که در دو سطح (به صورت خصوصیات و زیر خصوصیات) ارائه شد. این خصوصیات عبارتند از :

- Functionality: مجموعه‌ای از خصوصیات که مربوط به وجود مجموعه عملکردها و خصوصیات مشخص آنها می‌باشد.

- Suitability: خصوصیت عملکردی مناسب باشد. به تناسب و سازگاری کارکردهای سیستم برمی گردد.
- Accurateness: این خصوصیت مربوط به درستی عملکرد می‌باشد. مثلا دستگاه **ATM** می‌تواند عملکرد توزیع پول نقد را داشته باشد ولی آیا این عملکرد نتیجه درستی دارد؟
- Interoperability: عموما یک سیستم یا مؤلفه نرم افزاری بصورت ایزوله عمل نمی‌کند. این خصوصیت توانایی و قابلیت تعامل مؤلفه ها با یکدیگر را بیان می کند.
- Compliance: بر توانایی تطابق و مطلوبیت نرم افزار نظارت دارد. (این‌که نرم افزار مطابق با قواعد و راهبردهای آن صنعت یا مؤسسه باشد).
- Security: این خصوصیت به دسترسی غیرمجاز از عملکردهای سیستم برمی گردد.

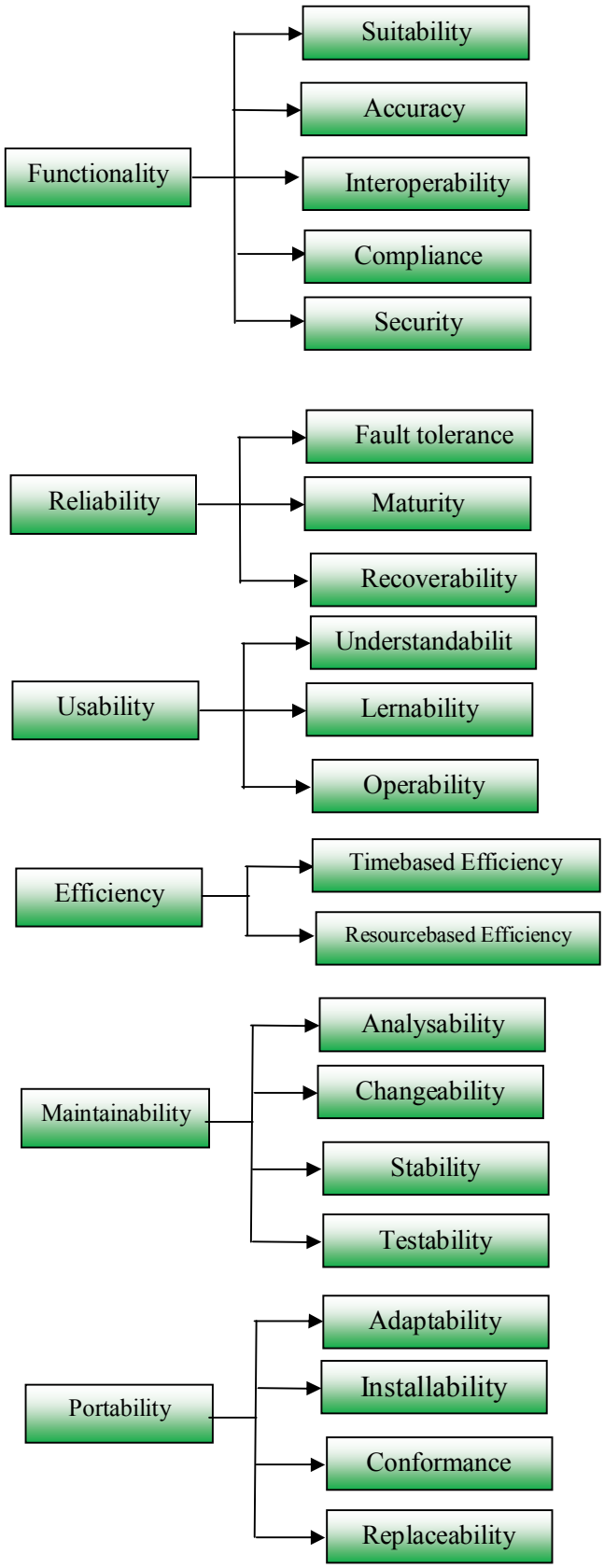
- Reliability : مجموعه ای از خصوصیات که مربوط به توانایی نرم افزار به حفظ سطح کارایی تحت شرایط تعیین شده و در پریود زمانی تعیین شده میباشد.
  - Maturity: این خصوصیت بر فراوانی خطاهای موجود در نرم افزار دلالت دارد.
  - Fault tolerance: توانایی نرم افزار در تحمل و ترمیم خطاهای سیستمی و محیطی می‌باشد.
  - Recoverability: توانایی برگرداندن سیستم به زمان قبل از خطا و بازآوری کلیه ارتباطات شبکه ای و داده‌ها.
- Usability: مجموعه ای از خصوصیات که مربوط به استفاده از نرم افزار می‌باشد.
  - Understandability: تعیین سهولت اینکه عملکرد سیستم به راحتی قابل فهم باشد. به مدل های ذهنی کاربر در روش‌های تعامل کاربر و کامپیوتر[۲۳] برمی گردد.
  - Learnability: به میزان کوشش برای یادگیری برای کاربران مختلف اشاره دارد. مانند کاربران مبتدی،حرفه ای و ...
  - Operability: توانایی نرم افزار در کار کردن با یک کاربر تعیین شده و در یک محیط تعیین شده می‌باشد.
- Efficiency: مجموعه ای از خصوصیات که مربوط به ارتباط بین سطح کارایی و مقدار منابع استفاده شده تحت شرایط تعیین شده می‌باشد.

[۲۳] Human Computer Interaction methods

- Time behavior: زمان پاسخ برای یک محصول معین را مشخص می‌کند. مانند سرعت اجرای تراکنش
- Resource behavior: منابع استفاده شده را مشخص می کند. مانند حافظه، پردازنده، دیسک شبکه

- Portability: مجموعه‌ای از خصوصیات که مربوط به توانایی نرم افزار جهت انتقال از یک محیط به محیط دیگر است.
  - Adaptability: مشخص کردن توانایی سیستم جهت تغییر خصوصیات یا تغییر به یک محیط عملیاتی جدید.
  - Installability: مشخص کردن تلاش مورد نیاز برای نصب و راه اندازی نرم افزار.
  - Conformance: مانند خصوصیت compliance برای unctionalityمی‌باشد. منتها در مبحث قابلیت جابجایی.
  - Replaceability: جنبه‌های plug and play را در مؤلفه‌های نرم افزاری تعیین می‌کند و اینکه چقدر راحت است که مؤلفه‌های نرم افزاری مطابق با محیط عوض شوند.
- Maintainability: مجموعه‌ای از خصوصیات که مربوط به توانایی نرم افزار جهت اعمال تغییرات و اصلاحات درخواست شده می‌باشد.
  - Analyzability: توانایی تشخیص دلیل اصلی خرابی در یک نرم افزار را تعیین می‌کند.
  - Changeability: میزان تلاش جهت تغییر یک سیستم می باشد.
  - Stability: حساسیت یک سیستم در مقابل تغییر می باشد. (که ممکن است تغییر، تأثیرات منفی و ناخواسته‌ای را به همراه داشته باشد).
  - Testability: تلاش مورد نیاز برای بررسی و تست سیستم تغییر داده شده می‌باشد.

یکی از تفاوت‌های کلیدی و اصلی این مدل کیفیتی با سایر مدل‌ها در این بود که در آن هر زیر خصوصیت تنها توسط یک خصوصیت در سطح بالاتر تحت تأثیر قرار می‌گرفت، در حالی‌که در سایر مدل‌ها اینگونه نبود.


Functionality
Suitability
Accuracy
Interoperability
Compliance
Security
Reliability
Fault tolerance
Maturity
Recoverability
Usability
Understandabilit
Lernability
Operability
Efficiency
Timebased Efficiency
Resourcebased Efficiency
Maintainability
Analysability
Changeability
Stability
Testability
Portability
Adaptability
Installability
Conformance
Replaceability

۲-۲-٤ مدل FURPS :

از دیگر مدل‌های کیفی ارائه شده می‌توان به مدل پیشنهادی **Grady** که**FURPS** نامیده می‌شود اشاره نمود. این مدل شامل **F**unctionality **U**sability, **R**eliability, **P**erformance, **S**upportability می‌باشد. که تعاریف آن عبارتند از:

- قابلیت عملکرد: که می تواند شامل امکانات ،قابلیت‌ها و امنیت باشد.
- قابلیت استفاده: که می‌تواند شامل فاکتورهای انسانی، زیبایی[۲۴]، توافق و سازگاری در واسط کاربر، راهنمای برخط و حساس به متن، مستندات کاربر و اصول آموزش می‌باشد.
- قابلیت اطمینان: که می‌تواند شامل تکرار و شدت خطاها، قابلیت ترمیم، قابلیت پیش‌بینی، صحت و درستی و متوسط زمان رخداد خطا می‌باشد.
- کارایی: شرایطی را از قبیل سرعت،کارایی، در دسترس بودن، صحت و درستی، توان عملیاتی، زمان پاسخ، زمان ترمیم و بهره‌وری منابع را بر نیازمندی‌های کیفی تحمیل می کند.
- قابلیت پشتیبانی [۲۵]: که می‌تواند شامل قابلیت تست، قابلیت توسعه‌پذیری[۲۶]، قابلیت انطباق[۲۷]، قابلیت نگهداری، سازگاری ، قابلیت تعمیرپذیری[۲۸]، قابلیت نصب[۲۹]

۲-۲-۵ مدل Kazman، Clements

در این دسته‌بندی صفات کیفی بر اساس اینکه در زمان اجرا قابل مشاهده هستند یا خیر، در دو دسته قرار می گیرند:

۱. صفات کیفیتی قابل مشاهده در زمان اجرا: این صفات نشان می دهند که در طول مدت اجرا، یک سیستم چقدر خوب می‌تواند نیازمندی‌های رفتاری خودش را تأمین کند. یعنی به لحاظ رفتاری معین می کند که آیا سیستم نتایج را برآورده می کند و آیا این نتایج را در زمان درست برآورده می‌سازد یا خیر؟در واقع این صفات کیفیتی مربوط به ویژگی‌های قابل

---

[۲٤] aesthetic
[۲۵] Supportability
[۲٦] extensibility
[۲۷] adaptability
[۲۸] serviceability
[۲۹] installability

مشاهده در حین اجرا هستند. یعنی نرم افزار را باید اجرا کرد تا مشخص شود که در اثر اجرای آن، چنین ویژگی‌هایی فراهم می شوند یا خیر؟

این صفات عبارتند از: کارایی[۳۰]، امنیت[۳۱]، در دسترس بودن[۳۲]، قابلیت عملکرد یا و ظیفه‌مندی[۳۳]، قابلیت کاربرد و استفاده[۳۴] استفاده[۳۴]

۲. صفات کیفیتی غیر قابل مشاهده در زمان اجرا: این صفات به‌گونه‌ای هستند که در زمان اجرا نمی‌توان تشخیص داد که به آنها دست یافته‌ایم یا خیر، البته باید به این موضوع توجه شود که ممکن است بعضی صفات در مرحله تحلیل، بعضی در مرحله طراحی و. . . خود را نشان دهند.

این صفات عبارتند از: قابلیت حمل[۳۵]، قابلیت استفاده مجدد[۳۶]، اصلاح پذیری[۳۷]، تجمیع پذیری[۳۸] و . . .

۳- صفات کیفیتی

انواع صفات کیفی چه قابل مشاهده در زمان اجرا و چه غیر قابل مشاهده در زمان اجرا، مربوط به خود سیستم و برنامه کاربردی هستند و به آنها صفات کیفیتی نیز می‌گویند ولی یک سری ویژگی‌هایی تحت عنوان صفات کیفیتی تجاری از آنها یاد می‌شود که در معماری اثرگذار هستند مانند طول عمر پیش بینی شده برای سیستم، زمانبندی جهت ساخت یک نسخه از محصول، زمان مورد نیاز برای فروش و هزینه و سود آوری .

در واقع خط مشی های مختلفی در مورد کیفیت نرم افزار وجود دارد که عبارتند از :

- برای سیستم‌های **Real time** و با ظرفیت پردازش بالا: **Performance**
- برای سیستم‌های قابل اطمینان و تحمل‌پذیر در برابر خرابی: **Dependability**
- برای سیستم‌های با کارکرد سیستم های انسانی: **Usability**

---

[۳۰] Performance
[۳۱] Security
[۳۲] Availability
[۳۳] Functionality
[۳۴] Usability
[۳۵] Portability
[۳۶] Reusability
[۳۷] Modifiability
[۳۸] Integritability

- برای سیستم‌هایی که در مخاطره هستند و سیستم‌های با امنیت : **Safety**
- برای سیستم‌های امنیتی مثل بانک ها ودانشگاه‌ها: **Security**
- اکثر سیستم‌های موجود: **Integrity , Modifiability**

۳-۱کارایی[۳۹]:

کارایی ریشه در خاصیت منابع استفاده شده جهت برآوردن نیازها و همچنین چگونگی به اشتراک گذاری منابع در زمان مواجه با درخواست‌های چندگانه که باید روی یک منبع یکسان انجام شود، دارد. این نوع مسائل تحت عنوان مسائل زمانبندی مطرح می‌شود. کارایی صفتی است که به پاسخگویی سیستم مربوط می شود.

مسئله زمانبندی می‌تواند با استفاده از چهار نوع اطلاعات متفاوت بیان شود:

۱. Job ها و عملیات پردازش شده
۲. تعداد و نوع ماشین‌ها
۳. دیسیپلین‌هایی که شرایط و محدودیت‌هایی جهت تخصیص ایجاد می کنند.
۴. شرایطی که بتوان زمانبندی را ارزیابی کرد.

در مسائل زمانبندی به دوصورت می‌توان کارایی را محاسبه کرد، به این صورت که یا مدت زمان لازم برای پاسخگویی به یک رویداد را اندازه گرفت و روش دوم اینکه تعداد task های انجام شده دریک واحد زمانی را اندازه بگیریم.

۳-۲ امنیت[۴۰]

امنیت ویژگی از سیستم است که توانایی یک سیستم در مقاومت در برابر دسترسی‌های غیرمجاز و همچنین امتناع از ارائه سرویس در زمانی که در حال ارائه سرویس به کاربران مشروع می‌باشد را نشان می‌دهد. تلاش جهت تجاوز به امنیت، حمله[۴۱] نامیده می‌شود و اشکال مختلفی دارد. می‌تواند یک تلاش غیرمجاز در دسترسی به داده یا دستکاری داده وحتی می‌تواند امتناع از ارائه سرویس به

---

[۳۹] Performance
[۴۰] Security
[۴۱] attack

کاربران مشروع باشد. حمله‌ها می‌توانند در موضوعات مختلفی مطرح شود، از سرقت پولی با انتقال الکترونیک تا دستکاری داده‌های حساس. تعریف امنیت به مقوله ای برمی‌گردد که درآن از security یاد شده است. عموماً تعریف امنیت در سه مقوله کاربرهای دولتی و نظامی، کاربردهای مالی و بانکی و کاربردهای علمی و آکادمیک تعریف می‌شود.

- **در برنامه‌های دولتی و نظامی**: در چنین برنامه‌هایی، لو رفتن و افشای اطلاعات ریسکی جدی و پرخطر است که می‌تواند به طور کامل ارزش آن برنامه را از بین ببرد، ملاک اصلی در این کاربردها سعی در محرمانگی اطلاعات می باشد. در اینگونه کاربردها برنامه ها و سیستم عامل ها باید به گونه‌ای تولید شوند که سخت افزار و نرم افزار و از افشای اطلاعات جلوگیری نمایند.
- **در برنامه‌های بانکی و مالی**: در عملیات بانکی، سرمایه‌گذاری و محاسبات در ارتباط با کسب وکار مبحث امنیت بر روی حفاظت دارایی‌ها و اموال تأکید دارد. با آنکه افشای اطلاعات ریسک بزرگی است ولی ریسک بزرگ‌تر از آن دسترسی غیرمجاز به اطلاعات و اعمال تغییرات روی آنها می‌باشد. در این مقوله به حفاظت از جامعیت اطلاعات بیشتر توجه می شود.
- **در برنامه‌های آکادمیک و علمی:** در این برنامه‌ها مبحث اصلی حفاظت در ارتباط با حفاظت جهت جلوگیری از دسترسی غیرمجاز به منابع می باشد. زیرا منابع یا بسیار پرهزینه و گران قیمت هستند و یا منابع بحرانی محسوب می‌شوند.

تعریف عمومی از امنیت به چند صورت آورده شده است:

۱. عاری از خطر (ایمنی)
۲. حفاظت از اطلاعات در برابر افشا، دستکاری و خرابکاری. حفاظت‌ها می‌تواند بصورت تکنیکی و اجرایی باشد.
۳. خصیصه ای است که منجر می شود سیاست امنیت با درجه‌ای از تضمین محقق شود.
۴. معمولاً در جهت محرمانه سازی با هدف ایجاد اعتماد، بخصوص در شرایطی که لایه هایی چندگانه امنیتی نیاز می‌باشد.

۳-۳ در دسترس بودن[۴۲]

این صفت با خرابی و نقص سیستم و نتایج مرتبط با آن سرو کار دارد. خرابی زمانی رخ می‌دهد که سیستم قادر به تحویل سرویسی پایدار بر اساس ویژگی‌ها و خصوصیاتش نباشد. باید به این نکته توجه داشت که خرابی با نقص متفاوت است. در صورتیکه که نقص تصحیح یا پنهان نشود به خرابی بدل می‌شود و توسط کاربر سیستم قابل روئیت نمی‌باشد ولی خرابی توسط کاربر سیستم مشاهده می‌شود. در بحث دسترس پذیری سیستم باید به این موارد پاسخ داده شود که چگونه خرابی سیستم تشخیص داده می‌شود؟در زمان خرابی چه مشکلی پیش می‌آید؟سیستم می‌تواند حداکثر چه مدت زمان خارج از سرویس باشد؟ چگونه می‌توان از خرابی جلوگیری کرد؟ و اینکه چه نوع اخطاری در زمان بروز خرابی ارائه شود؟مهمترین موضوع در زمان رخداد خرابی این است که سیستم چه وقت تعمیر می‌شود. به عبارت دیگر دسترس پذیری احتمال عملیاتی بودن سیستم در زمان نیاز می‌باشد. یعنی در دسترس بودن مرتبط با کسری از زمان است که سیستم زنده است و اجرا می شود.

$$\alpha = \frac{\mathbf{MinTime - to - Failure}}{\mathbf{MinTime - to - Failure + MinTime - to - Repair}}$$

$\mathbf{MinTime - to - Failure}$ میانگین زمان خطا و $\mathbf{MinTime - to - Repair}$ میانگین زمان اصلاح می‌باشد. در دسترس بودن رابطه نزدیکی با قابلیت اطمینان دارد، یعنی هر چه قابلیت اطمینان در سیستمی زیاد باشد آن سیستم در دسترس‌تر خواهد بود و بالعکس. قابل اطمینان بودن یعنی توانایی سیستم برای ادامه عملیات در طول زمان.

پارامترهایی که در این خصوصیت دخالت دارند، هر دو وابسته به معماری می باشند. میانگین زمان خطا عمدتًا با ایجاد یک معماری با تحمل خطای بالا، بیشتر می شود، حال آنکه تحمل خطا با تکرار عناصر پردازشی مهم و اتصالات در معماری بدست می آید. پارامتر بعدی متوسط زمان اصلاح است که هر چه معماری بهتر باشد، متوسط زمان وقوع خرابی بالاتر می‌رود و اضافه کردن عناصر پردازشی برای وقتی که بحرانی اتفاق می افتد، روی معماری اثر می‌گذارد.

[۴۲] Availability

۳-٤ قابلیت استفاده[٤٣]

قابلیت استفاده به این مطلب موضوع می‌شود که واسط کاربر تا چه حد برای استفاده کاربرساده می باشد و نیازهای او را برآورده می‌سازد. قابلیت استفاده بخشی ازسودمندی[٤٤] است وشامل موارد زیر می‌باشد:

قابلیت استفاده مربوط به این است که یک کاربر به چه میزان به راحتی قادر است کار مورد نظر خود را انجام دهد و همچنین سیستم چه میزان از انواع خواسته‌های آن را پشتیبانی می‌نماید. لذا اگر بخواهیم قابلیت استفاده یک محصول نرم افزاری را اندازه گیری نمائیم باید مسائل زیر را بررسی کنیم.

- **بازدهی**[٤٥]: زمانیکه کاربریاد گرفت با سیستم کار کند ،چقدرسریع می تواند کارها را به انجام برساند(سرعت انجام عملیات مورد درخواست کاربر).
- **قابلیت به خاطر سپاری**[٤٦] : زمانی‌که کاربر پس ازیک بازه زمانی وقفه مجدداً به سیستم برمی‌گردد،چقدر راحت می‌تواند عملیات را به یاد آورد؟
- **خطاها**[٤٧] : آیا امکان دارد در سیستم کاربردچار خطا شود ؟این خطاها چقدر است و سیستم را با چه مشکلاتی روبرو می‌کند؟آیا مکانیزم هایی برای مقابله و یا اصلاح خطا درنظر گرفته شده است؟
- **رضایت** : آیا کاربر از کار با سیستم احساس رضایت و خوشنودی دارد؟
- **قابلیت یادگیری**[٤٨] : سادگی و مدت زمانیکه طول می‌کشد تا کاربری که به تازگی با سیستم روبرو شده است بتواند عملیات اولیه و اساسی سیستم را یاد بگیرد .

قابلیت استفاده نقش مهمی را در کل فرایند طراحی سیستم بازی می‌کند. در زیر مراحل اصلی معرفی می‌شود:

- قبل ازشروع طراحی جدید، طراحی قدیمی را بررسی کرده و روی بخش‌های خوب آن تأکید کرده وبخش‌های بد آن را مد نظر داشته باشید.

---

[٤٣] Usability
[٤٤] usefulness
[٤٥] Efficiency
[٤٦] Memorability
[٤٧] Errors
[٤٨] Learnability

- سعی کنید شناخت دقیقی ازکاربر بدست آورید. (گاهی پرداختن به این مورد نیازبه ورود به فیلد علمی جدیدی دارد).
- طراحی خود را بارهاو بارها با توجه به ایده ها ، guideline ها و مطالعات مرتبط، تست و بررسی کنید.
- در مرحله پیاده‌سازی به قابلیت استفاده بسیار توجه داشته باشید ،زیرا تغییر در این مرحله بسیارپر هزینه میباشد.

۳-۵ تغییر پذیری[۴۹]

تغییر پذیری، توانایی سیستم جهت تحمل تغییرات و تعمیرات انجام گرفته در سیستم می‌باشد. MTTR یک معیار کمی برای قابلیت تغییر پذیری می باشد، اما تمام موارد را بیان نمی‌کند. برای مثال، بعضی از سیستم‌ها توسط کاربران و بعضی توسط سازنده و بعضی توسط هردو پشتیبانی می شوند. فرض کنید ماشینی خرابی یک برد را تشخیص می‌دهد، پیغامی را به شرکت سازنده ارسال می‌کند و سپس شرکت برد تعویضی را با دستورالعمل نصب و نحوه جایگزنی ارسال می‌کند در این مثال علاوه برMTTR مورد هزینه نیز وارد می-شود.

امکان تغییر در سیستم در موارد زیر می‌تواند رخ دهد:

- تصحیح خطاها و نواقص پروژه
- درخواست نیازمندی های جدید درسازمان و برآوردن این نیازمندی‌ها(اضافه کردن یک کارایی جدید، بهبود کارایی، اصلاحات، ایجاد ساختارهای جدید).
- انجام پشتیبانی آسان تر درآینده.
- انجام تغییرات جهت تطبیق با تغییر محیط(لذا قابلیت حمل می‌تواند به عنوان جزئی از تغییر پذیری مطرح شود).

تغییر پذیری در دو مسئله بررسی می‌شود:

۱. چه چیزی می‌تواندتغییرکند؟ تغییر می‌تواند درهر جنبه سیستم رخ دهد مثلاً می‌تواند تغییر در توابعی که در سیستم محاسبه می‌شود نباشد، می‌تواند بستر(patformی) که سیستم در آن ساکن است(سخت افزار،سیستم عامل،میان افزار و . .) تغییر کند یا محیطی که سیستم در آن اجرا می‌شود (محیطی که سیستم در آن تعامل دارد، پروتکل‌هایی که با استفاده از آن با جهان

[۴۹] Modifiability

خارج ارتباط برقرار می‌کند و . . .)، یا خصوصیات کیفی که سیستم ارائه می کند (کارایی،قابلیت اطمینان و حتی تغییرات و . . .) یا ظرفیتش (تعداد کاربری که پشتیبانی می کند، تعداد عملیاتی که بطور همزمان انجام می‌دهد و. . .). بعضی از بخش-های سیستم مانند واسط یا بستر بر اساس تغییرات انجام شده ممکن است تغییر کنند. (تغییرات انجام شده در platform قابلیت حمل نامیده می شود).

۲. چه زمانی تغییر ایجاد شده و چه کسی آن را انجام می دهد؟ عمدتاً درگذشته تغییرات روی کد اصلی انجام می‌شد که دراین صورت تولیدکننده باید تغییرات را اعمال، تست را انجام داده و یک release جدید را ارائه دهد. اما هم اکنون پرسش چه زمانی تغییرات اعمال میشودبا سوال چه کسی آن راانجام می دهد در ارتباط است. تغییرات می تواند در طی فاز پیاده سازی (با تغییر کد برنامه)، در طی فاز کامپایل (با استفاده از سوئیچ‌های compile-time) ، در طی فاز ساخت (با انتخاب کتابخانه‌ها)، در طی فازپیکربندی setup،در طی فازاجرا (توسط تنظیمات پارامتر) انجام شود. همچنین این تغییرات می تواند توسط کاربر نهایی، تولید کننده سیستم یا مدیر سیستم انجام شود.

در زمان اعمال تغییرات،کاربرد جدیدباید طراحی، پیاده سازی، تست و گسترش داده شود. تمام این عملیات هزینه و زمان می برد.

۳-٦ ثبات و پایداری[۵۰]:

توانایی سیستم نرم افزاری در اجتناب از تأثیرات ناخواسته درزمان تغییر و ویرایش سیستم نرم افزاری می باشد. در این جا ثبات به مفهوم رفتار پایدار از سیستم در زمان استفاده نمی باشد. ثبات به قابلیت تغییر[۵۱] برمی‌گردد. از این جهت که قابلیت تغییر اندک Stability کمتری را نیز دارد. این مسئله به آن حقیقت برمی گردد که تلاش جهت تغییر یک سیستم با قابلیت تغییر پایین منجر به ریسک بزرگی می شود، که می‌تواند منجر به خرابی سیستم شود. جهت داشتن سیستم‌هایی باخصوصیات ثبات و پایداری مناسب توصیه‌هایی شده به شرح ذیل شده است:

- داشتن قابلیت تغییر بالا در سیستم
- انجام ارزیابی پیشگویانه از تغییرات آینده

---

۵۰ Stability
۵۱ Changeability

- اجتناب از داشتن کلاس هایا مؤلفه های مرکزی به گونه ای که از بخش های دیگر برنامه به آن دسترسی وجود داشته باشد. (و یا حد الامکان جلوگیری از تغییر مؤلفه و کلاسهای مرکزی)

۳-۷ قابلیت همکاری [۵۲]:

قابلیت اینکه دویاچند سیستم بتوانند با یکدیگر اطلاعات را رد و بدل کنند و از این اطلاعات نیز استفاده کننده، می باشد. در مدل MC Call قابلیت همکاری به دو خصیصه تجزیه می شود:

- Communications commonality
- Data commonality

مفهوم اساسی ثبت شده در این دو خصیصه دست یافتن به قابلیت همکاری بالا میباشد. نیاز به خصیصه قابلیت همکاری باید در مراحل اولیه چرخه حیات سیستم موردتوجه قرارگیرد (قبل یا درحین عملیات طراحی درفرایند معماری سیستم). به طور نمونه معیارها و مکانیزم هایی برای پشتیبانی قابلیت همکاری به شرح ذیل می باشد:

- اطمینان از استفاده از پرتکل های سازگار ارتباطی داده، مدل های نمایش داده وساختارهای کنترلی مناسب.
- استفاده از سیستم های میان افزار موجود و مستقل از مکان.
- استفاده از استانداردهای باز(open)

خاصیت تجمیع پذیری[۵۳] نیز می تواند در کنار این ویژگی استفاده شود. این ویژگی نشان می دهد که مؤلفه هایی که جدا از هم ساخته شده اند چقدر می توانند به درستی با هم کار کنند. این ویژگی بستگی به چگونگی پنهان سازی پیچیدگی های مؤلفه‌ها، پروتکل های ارتباطی آنها دارد.

۳-۸ قابلیت حمل [۵۴]:

تعریف قابلیت حمل به اینگونه می باشد: " قابلیت اینکه یک سیستم یا مؤلفه بتواند به راحتی از یک محیط سخت افزاری یا نرم افزاری به محیط دیگری منتقل شود" . خصوصیاتی که این ویژگی را پشتیبانی می کنند عبارتند از:

---

[۵۲] Interoperability
[۵۳] Integrity
[۵۴] Portability

- سادگی [۵۵]
- استقلال از سیستم نرم افزاری
- استقلال از دستگاه[۵۶]
- کامل و جامع [۵۷].

موارد استقلال از سیستم نرم افزاری و استقلال از ماشین موارد مطلوبی برای رسیدن به خصوصیت قابل حمل می باشند ولی معمولا بطور کامل امکان‌پذیر نمی باشد. سیستم هایی که برای ارسال به platform های مختلف تهیه می‌شوند اغلب یک لایه انتزاعی سخت افزاری[۵۸] را در خود جای داده اند که سیستم را از platform محلی خود جدا می‌کند. مثالی از چنین سیستم هایی ویندوز های شرکت ماکروسافت می باشد که HAL های مختلف ،CPU های مختلفی را پشتیبانی می‌کند. جهت داشتن سیستم هایی باخصوصیات قابل حمل توصیه هایی شده است که به آن ذیلاً اشاره شده است:

- از اتصال سیستم به platformاجتناب کنید، در صورت لزوم سعی کنید انتزاعی[۵۹] از platform را بسازید.
- تاآنجا که ممکن است ازexternal sevices استفاده کنید.
- بطور واضح مسئولیت های مؤلفه ها را در معماری تعریف کنید.
- از زبان های برنامه نویسی استفاده کنید که حدالامکان، خودشان قابل حمل باشند.

۳-۹ قابلیت استفاده مجدد[۶۰]

به معنای آن است که ماژول نرم افزاری یا دیگر مؤلفه ها و تولیدات به گونه ای طراحی شوند که بتوانند در دیگر برنامه های محاسباتی یا سیستم های نرم افزاری استفاده شوند. این خصوصیت دارای زیر خصوصیاتی می باشدکه عبارتند از:

- سادگی
- کلیت[۶۱]

---

[۵۵] Simplicity
[۵۶] machine/device independence
[۵۷] self-contained
[۵۸] Hardware Abstraction Layer (HAL)
[۵۹] Abstraction
[۶۰] Reusability

- پیمانه ای بودن[۶۲]
- استقلال از سیستم نرم افزاری
- مستقل از ماشین

می توان گفت برای داشتن سیستم هایی با قابلیت استفاده مجدد حدالامکان سیستم راcomponent base ایجاد کرد. مفهوم سادگی به این معناست که عملکرد چنین سیستم هایی باید بصورت عمومی باشد. سیستم های ماژولار سیستم هایی هستند که عملکرد آنهادر ماژول‌های مجزایی ایزوله شده است. متضاد آن سیستم های مونولیتیک یا یکپارچه می‌باشد که تمام عملکرد ها در یک واحد مجزا جمع شده است. داشتن ماژول های مجزا می‌تواند عملکرد را ازسیستم جدا کند. جهت داشتن سیستم هایی باخصوصیات قابلیت استفاده مجدد توصیه هایی شده است که ذیلاً به آن اشاره گردیده است:

- ساختن راه حل های عمومی و ساده جهت افزایش قابلت فهم که و در نتیجه قابلیت استفاده مجدد نیز افزایش می یابد.
- سیستم بصورت ماژولار طراحی و پیاده سازی شود وتا حد امکان از پیوند ضعیف بین ماژول ها اطمینان حاصل گردد.
- سیستم عملیاتی خاص یا platform. سخت افزاری بخصوصی را دنبال نکنید.
- جهت ارتباطات از میان افزارهای استاندارد مانند CORBA یا COM/DCOM استفاده شود.
- واسط مؤلفه‌ها را سیستمیک، سازگار و همسان ساخته شوند.

۳-۱۰ قابلیت آزمایش[۶۳]

قابلیت آزمایش یک نرم افزار، توانایی ایجاد تسهیلاتی جهت تست معیارها و کارایی سیستم جهت بررسی درجه برآورده کردن نیازها وخواسته‌ها است. این ویژگی با خصوصیات زیر در ارتباط است :

- سادگی
- خود توصیف[۶۴]

---

[۶۱] Generality
[۶۲] Modularity
[۶۳] Testability
[۶۴] Self-descriptiveness

- استفاده از ابزار[۶۵]
- پیمانه‌ای و ساخت‌یافته
- دسترس پذیر
- جوابگو[۶۶]
- خبر رسان (آماده برای ارائه گزارش)[۶۷]

قرار دادن ویژگی قابلیت تست در سیستم های ساده به مراتب راحت‌تر از سیستم-های پیچیده می‌باشد. تنظیم ابزار به معنای داشتن کاوشگرهایی در سیستم جهت تست است. یک سیستم خودتوصیف، سیستمی مستند و قابل خواندن[۶۸] است. یک سیستم ماژولار، به ماژول های مجزایی تقسیم شده است که تست هر کدام از آنها بصورت ایزوله و مستقل ازدیگر درشرایط مورد نیاز انجام می‌شود. ویژگی ساخت‌یافته بودن بیان می‌کند که بخش‌های سیستم در وضعیت های پایداری سازمان یافته‌اند. جوابگویی به آن معناست که سیستم قادر است استفاده از کد را موردسنجش قرار دهد. بطور نمونه چنین سنجش‌هایی با ابزار پوششی انجام می شود که برای بسیاری از زبان های برنامه نویسی موجود است. خبر رسان به آن معناست که امکان مشخص کردن ورودی ها و خروجی های سیستم به راحتی انجام شود.

روش طراحی و پیاده سازی سیستم بر ویژگی قابلیت تست تاثیر می‌گذارد. بعضی از راهبردها برای ایجاد و افزایش قابلیت تست درسیستم عبارتند از:

- ساختن solution های ساده به گونه‌ای که تسهیلاتی را برای شرایط تست ایجاد کند.
- اطمینان از اینکه بخش‌های مختلف سیستم ساختیافته و ماژولار می‌باشد.
- سیستم بطورمناسب مستند شود.
- مکانیزم هایی ایجاد شود که شرایط کاوش کد را فراهم کند. مثلا مکانیزم های اشکال زدایی خروجی [۶۹]

---

[۶۵] Instrumentation
[۶۶] Accountability
[۶۷] Communicativeness
[۶۸] readable
[۶۹] Debug output

- ایجاد روش‌هایی جهت مشخص کردن راحت ورودی و قابل فهم کردن خروجی. از ایجاد ناهماهنگی در ورودی و خروجی اجتناب شود.
- واسط‌های ارتباطی و مؤلفه‌ای مجزا و واضحی تعریف گردد. مثلا استفاده از میان افزارهای استاندارد.

٤- تاثیر توسعه Component Base بر خصوصیات کیفی نرم افزار:

توسعه CB[٧٠] مفهومی است که توسعه دهندگان را قادر به ساختن سیستم های نرم افزاری که از بخش های قابل استفاده مجدد تشکیل شده اند، می‌کند. مهندسی مبتنی بر مؤلفه یک فیلد مورد بررسی در مهندسی نرم افزار است که بر مبنای تئوری های از بیش تعریف شده روی نرم افزار، معماری نرم افزار، چارچوب‌های نرم افزار و الگوهای طراحی نرم افز را و همچنین تئوری بزرگ برنامه نویسی شی‌گرا و طراحی شی‌گرا بنا شده است و ادعا دارد اجزای نرم افزاری (مؤلفه‌ها) می توانند قابلیت اطمینان بالاتر و قابلیت تغییر بیشتری را ایجاد کنند. رهیافت CBSD سازمان‌ها را از فعالیت تولید کاربرد به سمت مونتاژ کاربرد سوق داده است. هدف از این رویکرد کاهش زمان تولید، هزینه و تلاش مورد نیاز می باشد در حالی‌که محصول نهایی انعطاف پذیری، قابلیت استفاده مجدد و قابلیت اطمینان را نیز با خود دارد.

مؤلفه یک جزء مستقل و قابل تعویض از سیستم می باشد که یک عملکرد مشخص را درفضای یک معماری خوش تعریف برآورده می‌سازد. یک جزء درکی فیزیکی از یک مجموعه واسط را فراهم می‌کند و با آن مطابقت دارد.

٤-١ تاثیر CBD برخصوصیات کیفی نرم افزار :

برای اینکه توجیهی جهت تاثیر **CBD** بر خصوصیات کیفی نرم افزار ارائه کنیم باید ابتدا تعریف شود که چگونه مفاهیم مختلف بر یکدیگر تاثیر می‌گذارند. (CBD را هم به عنوان یک مفهوم داریم).

خصوصیات کیفی می‌توانند به سه گروه مختلف تقسیم شوند: (شکل ۴)

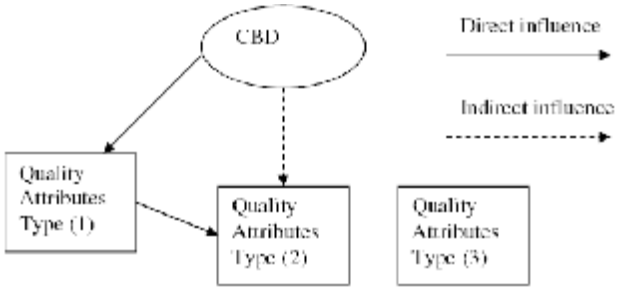
١. بعضی از خصوصیات هستند که CBD مستقیماً روی آن تأثیر دارد. (پذیرش تأثیر بصورت مستقیم)

---

[٧٠] Component Base

۲. دیگر خصوصیات لزوما توسط CBD تحت تاثیر قرار نمی‌گیرند، اما توسط دیگر خصوصیاتی که تحت تاثیر CBD هستند تحت تاثیر قرار می‌گیرند. (پذیرش تأثیر بصورت غیر مستقیم)

۳. بعضی از خصوصیات نه بطور مستقیم و نه بطور غیر مستقیم تحت تأثیر قرار نمی گیرند.



شکل ۴

از طرفی ارتباط بین خصوصیات کیفی می تواند بصورت های زیر باشد :

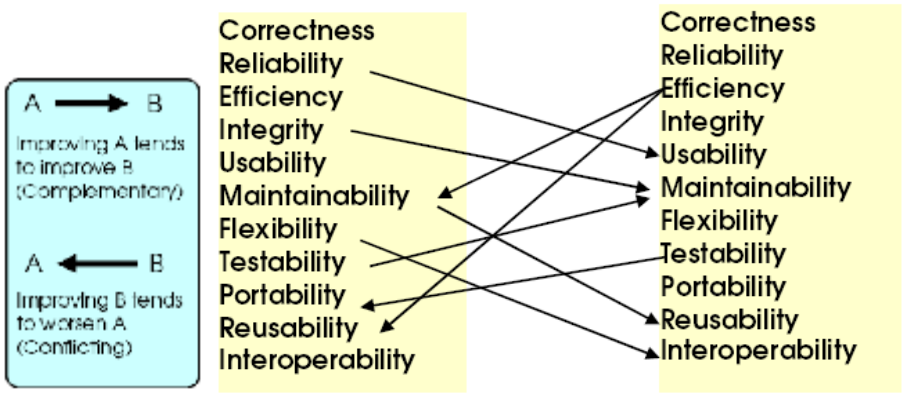
۱. بی تفاوت[۷۱] : یعنی بدون وابستگی.

۲. مکمل[۷۲]: یعنی X بهتر موجب Y بهتری می شود.

۳. مخالف و در تضاد [۷۳]: یعنی X بهتر موجب Y بدتری می شود.

شکل زیرمثال هایی از خصوصیات کیفی متضاد و مخالف را در مدل McCall نشان می‌دهد.

[۷۱] Indifferent
[۷۲] Complementary
[۷۳] Conflicting



شکل ٥

٤-٢ تاثیر CBD بر خصوصیات کیفی در مدل ISO 9126:

استفاده از توسعه مبتنی بر مؤلفه در بعضی از حالات مستقیماً بر روی خصوصیات کیفی در یک سیستم تاثیر می‌گذارد. مثلاً در مدل McCall از دیدگاه تحول محصول خصوصیاتی مثل قابلیت حمل، قابلیت استفاده مجدد و قابلیت همکاری وجود دارد. این خصوصیات به عنوان خصوصیات پشتیبانی شده توسط مفهوم توسعه مبتنی بر مؤلفه شناخته می شود و هر کسی متوجه می شود که استفاده از توسعه مبتنی برمؤلفه تأثیر مثبت و واضحی روی این خصوصیات دارد. مثال دیگر قابلیت استفاده مجدد می باشد که توسط تکنولوژی مبتنی بر مؤلفه تحت تاثیر قرار می گیرد. اگر تولید کننده نرم افزار از یک واسط کاربر مناسب و شناخته شده از کتابخانه مؤلفه [٧٤] استفاده کند، می تواند با استفاده از این واسط کاربر مناسب، usability را بالا برد.

ذیلاً چگونگی تأثیر CBD بر خصوصیات کیفی ISO 9126 را بیان شده است. در اینجا هم تأثیر خصوصیات مثبت و هم تأثیر خصوصیات منفی بیان شده است.

[٧٤] Component library

خصوصیت	تاثیر مثبت	تاثیر منفی
قابلیت عملکرد	استفاده از مؤلفه‌های از بیش موجود اجازه تحویل سریع تر و عملکرد بهتری را فراهم می کند.	مؤلفه‌های از بیش موجود ممکن است آن عملکردی را که شما نیاز دارید ندهند.
قابلیت نگهداشت	ساختار ماژولار راه حل مبتنی بر مؤلفه امکان جایگزینی راحتی را به مؤلفه های منحصر به فرد می دهد.	برای مؤلفه های black-box، شما قادر به انجام هرگونه تعمیر و پشتیبانی مؤلفهؤها را آنگونه که خود می خواهید ندارد.
قابلیت استفاده	استفاده از مؤلفه های استاندارد، GUI های عامه پسند را پشتیبانی می کند.	
کفایت و کارایی	گلوگاه های کارایی می تواند تعیین شده و سپس نیاز به تنظیم کارایی درتعدادی مؤلفه حیاتی متمرکز شود. همچنین مؤلفه‌ها می‌توانند بطور داخلی جهت ارتقاء کارایی بدون تاثیر بر خصوصیاتشان بهینه شوند.	نیاز به توسعه سیستم مطابق با یک چارچوب مؤلفه داده شده و استفاده از مؤلفه‌هایی که لزوماً یکدیگر را بهینه نمی کنند، می تواند برای کارایی منفی باشد.
قابلیت حمل	مشخصات مؤلفه ،استقلال آنها از Platform می‌باشد. بنابراین یک مؤلفه می تواند سریعا برای یک platform جدید بدون آنکه تاثیری بر دیگر مؤلفه ها داشته باشد، ساخته شود .	
قابلیت اطمینان	با داشتن یک سری مشخصات کامل،رسمی و معلوم از یک سری مؤلفه ها سوال درباره قابلیت اطمینان یک مؤلفه، آسان می شود. به این صورت که: آیا مؤلفه یک پیاده سازی کامل و صریح از خصوصیاتش است؟ قابلیت اطمینان یک کاربرد عموما یک موضوع جداگانه می باشد. ولی بطور واضح قابلیت اطمینان در زمانی‌که از مؤلفه های قابل اطمینان استفاده میشود، افزایش می یابد.	

جدول ا: تأثیر CBD بر خصوصیات کیفی ISO 9126

می توان با پالایش و سفارشی کردن استاندارد ISO9126 برای خصوصیات کیفی نرم افزار آن را به گونه‌ای تغییر دهیم که خصوصیاتی کیفی برای سیستم های مبتنی بر مؤلفه و بالاخص برای مؤلفه های COTS [۷۵] قابل استناد باشد.

خصوصیات	زیر خصوصیت (زمان اجرا)	زیر خصوصیت (چرخه حیات)
قابلیت عملکرد (Functionalability)	Accuracy Security	Suitability -Interoperability -Compliance
قابلیت اطمینان (Reliability)	Recoverability	Maturity
قابلیت استفاده (Usability)		Learnability -understandability- operability
کارایی و کفایت (Efficiency)	Time Behavior Resource Behavior	
قابلیت نگهداری (Maintainability)		Changeability Testability
قابلیت حمل (Portability)		Replaceability

جدول ۲: تقسیم بندی زیر خصوصیت در ISO 9126

- **قابلیت استفاده**: این خصوصیات به همراه تمام زیرخصوصیاتش می توانند بهترین مثال از خصوصیاتی باشند که معنای کاملا متفاوتی را برای مؤلفه‌های نرم افزاری دارند. دلیل این امر آن است که در CBSD کاربران نهایی مؤلفه ها، توسعه

[۷۵] Commercial Off-The-Shelf : محصول سخت افزار یا نرم افزار تجاری با تولید انبوه

دهندگان کاربرد و طراحان کاربرد می باشند. بنابراین قابل استفاده بودن یک مؤلفه به مفهوم توانایی و قابلیت آن مؤلفه برای ساخت سیستم یا نرم افزار توسط توسعه دهندگان می باشد.

- **کارایی:** همان مفهوم کارایی است که در استاندارد ISO 9126 مطرح شده است.

- **قابل نگهداشت:** این خصوصیت توانایی محصول نرم افزاری جهت ویرایش و تغییر می باشد. تغییرات شامل تصحیح، ارتقاء یا انطباق با نرم افزار بر اساس تغییرات در محیط، در نیازمندی ها یا در خصوصیات عملکردی می باشد. کاربر مؤلفه (مانند تولید کننده کاربرد) نیاز به انجام تغییرات داخلی ندارد اما نیاز به انطباق با آن، پیکربندی مجدد و تست مؤلفه قبل از پایان فرآیند تولید محصول، دارد. بنابراین قابلیت تست و قابلیت تغییر دو زیر خصوصیت می باشد که باید برای مؤلفه اندازه گیری شود.

- **قابلیت جابجایی**: توانایی انتقال محصول از یک محیط به محیط دیگر می باشد. . در BSD، خاصیت جابجایی یک ویژگی باطنی می باشد. بر اساس طبیعت مؤلفه است که اصولا برای استفاده مجدد در محیط های مختلف طراحی و تولید شده است.

معیارهایی که برای اندازه گیری این خصوصیات بکار برده می شود عبارتند از :

- وجود[۷۶]: معیاری که مشخص می کند که یک ویژگی در مؤلفه وجود دارد یا خیر ؟. یک مقدار بولین می باشد.

- زمان[۷۷] : این معیار برای اندازه گیری بازه های زمانی بکار برده می شود. یک مقدار عددی است که به همراه یک رشته جهت نمایش ثانیه/دقیقه /ساعت استفاده می‌شود.

- سطح[۷۸]: این معیار برای نشان دادن درجه توانایی و قابلیت استفاده می شود.

- نرخ[۷۹]: این معیار برای اندازه گیری درصد استفاده می شود.

---

[۷۶] Presence
[۷۷] Time
[۷۸] Level
[۷۹] Ratio

٤-٣ خصوصیت های قابل اندازه گیری در زمان اجرا :

این خاصیت ها به همراه معیارهای اندازه گیری آنهادر جدول زیر نشان داده شده است :

Sub-characteristic	Attribute	Type
Accuracy	*1. Precision* *2. Computational Accuracy*	*Ratio* *Ratio*
Security	*3. Data Encryption* *4. Controllability* *5. Auditability*	*Presence* *Presence* *Presence*
Recoverability	*6. Serializable* *7. Persistent* *8. Transactional* *9. Error Handling*	*Presence* *Presence* *Presence* *Presence*
Time behavior	*10. Response time* *11. Throughput* *12. Capacity*	*Time* *Integer* *Integer*
Resource behavior	*13. Memory utilization* *14. Disk utilization*	*Integer* *Integer*

جدول ۳: زیر خصوصیت ها و معیارهای آنان

٤-٤ خصوصیت های قابل اندازه گیری در طول چرخه حیات :

این خصوصیات به همراه معیارهای اندازه گیری آنهادر جدول زیر نشان داده شده است :

Sub-characteristic	Attribute	Type
Suitability	1. Coverage 2. Excess 3. Service Implementation Coverage	Ratio Ratio Ratio
Interoperability	4. Data Compatibility	Presence
Compliance	5. Standardization 6. Certification	Presence Presence
Maturity	7. Volatility 8. Evolvability 9. Failure removal	Time Integer Integer
Learnability	10. Time to use 11. Time to configure 12. Time to admin 13. Time to expertise	Time Time Time Time
Understandability	14. User Documentation 15. Help System 16. Computer Documentation 17. Training 18. Demonstration Coverage	Level Level Presence Presence Ratio
Operability	19. Provided Interfaces 20. Required Interfaces 21. Complexity Ratio 22. Effort for operating 23. Tailorability 24. Administrability	Integer Integer Index Level Level Level
Changeability	25. Customizability 26. Customizability Ratio 27. Change Control Capability	Integer Index Level
Testability	28. Start-up Self-test 29. Tests Suite Provided	Presence Presence
Replaceability	30. Backward Compatibility	Presence

جدول ٤: زیر خصوصیت ها و معیارهای آنها

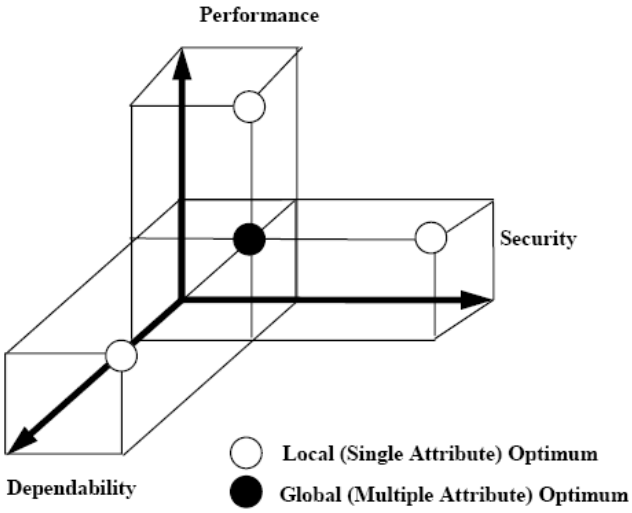
۵- مصالحه بین صفات کیفی :

اغلب سیستم ها برای برآوردن نیازهای کاربران fail می شوندکه بیشتر به دلیل نبودن کیفیت است و عمدتاً زمانی است که طراحان به دقت در برآورد بعضی نیازها بدون توجه به تأثیر آن بر دیگر نیازمندی ها عمل می کنند.

اصو لا رسیدن به یک کیفیت بر روی کیفیت دیگر تأثیر گذار است. مثلا ممکن است وقتی که کارایی را بالا ببریم، امنیت را پائین آوریم. یعنی ممکن است که همه صفات کیفی را نتوانیم با هم برآورده سازیم. دلیل اینکه نمی توانیم تمام صفات کیفی را در کنار هم داشته باشیم به این مسئله برمی گردد که گاهی بعضی از صفات ذاتاً با یکدیگر در تضاد هستند و مورد دیگر اینکه ممکن است خصوصیات کیفی در دید افراد مختلف متفاوت باشد. مثلا مفهوم امنیت می تواند در دید یک نفر به معنای رمز کردن اطلاعات[۸۰] و در دید دیگری به معنای Fire Wall باشد. مثلا فرض کنید امنیت و تحمل پذیری خطا بصورت انحصاری قابل دستیابی باشند ولی اگر

[۸۰] encryption

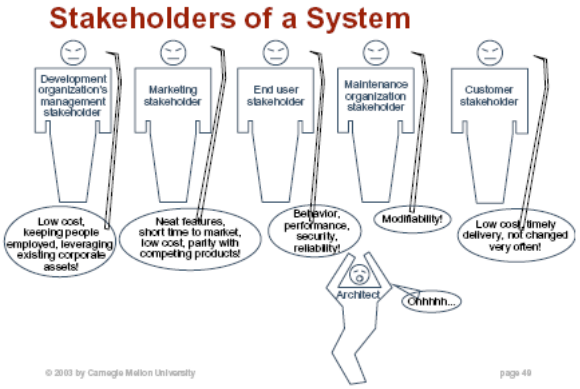
توأم با هم بخواهیم آنها را داشته باشیم باید مصالحه انجام دهیم و یکی از مسائل مطرح در ارتباط با بحث معماری و ویژگی‌های کیفی همین مصالحه است. بدین معنی که ممکن است همه صفات کیفی را نتوانیم همزمان فراهم آوریم، ولی باید ببینیم که کدام یک از این شرایط برای ما از اهمیت بیشتری برخورداراست. یعنی مثلا کارایی اهمیت بیشتری دارد یا قابلیت استفاده مجدد و به همین ترتیب در مورد بقیه صفات کیفی. در شکل این مفهوم برای سه ویژگی کیفی نشان داده شده است.



شکل ۷: مصالحه بین صفات کیفی

در مورد خصوصیات کیفی و تداخل احتمالی آنها می توان به موارد زیر اشاره کرد:

- همه خصوصیات به یک سیستم مرتبط نیستند و گاهی بعضی آنها را می توانیم در نظر نگیریم.
- به کارگیری خصوصیات بر اساس اهمیت آنها می باشد.
- تنها خصوصیاتی که در حد خاصی قرار دارند قابل بررسی و ارزیابی اند.
- تکنیک های ارزیابی که برای خصوصیات مهم به کار گرفته می شوند باید قابل سنجش و ارزیابی باشند.



شکل ۸: مصالحه بین خصوصیات مختلف چگونه ممکن است؟

مسئله تقدم و تاخر خصوصیات کیفی نیز به مسئله مصالحه بین خصوصیات کیفی برمی گردد . بعضی از خصوصیات کیفی مورد نظر کاربر و بعضی از خصوصیات کیفی مورد توجه تولید کننده می باشد. جدول زیر این خصوصیات را نشان می دهد.

Attribute	Interest
Reliablity	User
Availability	User
Robustness	User
Integrity	User
Flexability	User
Usability	User
Interoperatability	User
Efficiency	User
Testability	Developer
Maintainability	Developer
Reuseability	Developer
Portability	Developer

جدول ۵: خصوصیت ها و ارجاع کنندگان به آن

٦- (SQW) Software Quality Workshop

QAW روشی است که ذینفعان (stakeholder ها) آن را جهت کشف خصوصیات کیفی یک سیستم، در چرخه خصوصیات کیفی یک سیستم بکار میگیرند. QAW قبل از ایجاد معماری نرم افزار بکار گرفته می شود. بطور خلاصه QAW راهی جهت کشف، مستندسازی و اولویت بندی خصوصیات کیفی

در اوایل چرخه حیات سیستم بکار گرفته می شود. اهمیت توجه به این مسئله از آنجا ناشی می شود که ویژگی های کیفی، مشخصه سیستم، شرایط کاری سیستم و درایورهای طراحی معماری را تعیین می کنند . QAW کمک می کند تا حداکثر اطلاعات کیفی سیستم حاصل شود، همچنین موجب می شود تا ساختار سیستم تعیین شود.

٦-١ مراحل QAW:

QAW نیاز به همکاری و مشارکت stakeholder های سیستم دارد و حضور کلیه آنها در طول این زمان ضروری می باشد. QAW درگیر گام های زیر می باشد:

١. ارائه QAW و مقدمات [٨١]
٢. ارائه ماموریت /کسب و کار [٨٢]
٣. ارائه طراحی معماری [٨٣]
۴. تعریف محرک های معماری [٨۴]
۵. تبادل اندیشه [٨۵]
۶. تحکیم و یکپارچگی سناریو [٨۶]

[٨١] QAW Presentation and Introductions
[٨٢] Business/Mission Presentation
[٨٣] Architectural Plan Presentation
[٨٤] Identification of Architectural Drivers
[٨٥] Scenario Brainstorming
[٨٦] Scenario Consolidation

۷. اولویت بندی سناریو [۸۷]

۸. پالایش سناریو [۸۸]

٦-١-١ ارائه QAW و مقدمات

در این گام تسهیل کننده (facilitator) انگیزه QAW و هر گام آن را توصیف می دهد.

٦-١-٢ ارائه مأموریت /کسب و کار :

پس از مرحله اول، نماینده stakeholderها، مأموریت و کسب و کار خود را ارائه می دهد و نیازمندی های عملکردی سطح بالا، محدودیت ها و نیازمندی های کیفی نرم افزار را بیان می نماید.

٦-١-٣ ارائه طراحی معماری:

در صورتی‌که معماری تفضیلی سیستم موجود نباشد ممکن است توصیفات سیستم سطح بالا، از طریق ترسیم مفهوم یا دیگر ابزارها و فرآورده‌هایی ایجاد شود که بعضی از تکنیک های سیستم را به تفضیل توصیف می نماید. در این شرایط یک stakeholder تکنیکی و فنی، طراحی معماری سیستم را ارائه خواهد کرد. اطلاعات اولیه ارائه شده در این بخش می تواند شامل موارد زیر باشد:

- برنامه ها و استراتژی هایی به این منظور که چگونه نیازمندی های کلیدی ماموریتی و کسب و کار را برآورده کند.
- نیازمندی های کلیدی فنی و محدودیت ها از قبیل سخت افزار ،نرم افزار ،میان افزار و استانداردها مشخص می شود که در واقع تصمیمات معماری از آن استخراج می شود.
- دیاگرام های موضوعی موجود، دیاگرام های سیستمی سطح بالا و دیگر توصیفات نوشته شده ثبت و جنبه های کلیدی آن ارائه می شود.

٦-١-٤ تعریف محرک های معماری:

[۸۷] Scenario Prioritization
[۸۸] Scenario Refinement

در طول گام دوم و سوم، facilitator، اطلاعات مربوط به درایورهای معماری را که کلید اصلی فهم اهداف خصوصیات کیفی در سیستم می باشد را ثبت می کند. معمولاً این درایورها شامل نیازمندی های سطح بالا ،Concern های مأموریتی/کسب وکار ، مقاصد و اهداف و خصوصیات کیفی مختلف می باشد. در این گام سعی می‌شود که اطلاعات روشن شود، ممکن است اضافه، حذف یا ویرایشی صورت گیرد. لیست نهایی درایورهای معماری به stakeholder ها در طول سناریو تبادل اندیشه کمک می کند.

٦-١-٥ تبادل اندیشه :

پس از آنکه درایورهای معماری تعریف شدند، facilitator ، فرایند تبادل اندیشه را آغاز می‌کند که در این راه stakeholder ها سناریو را تولید می کنند. Facilitator ، بخش های خوب یک گزارش را مرور کرده و اطمینان حاصل می کند که سناریو در طول کارگاه بخوبی شکل گرفته است. هر stakeholder یک سناریو را بیان می کند و concern هایش را با توجه به سیستم ارائه می نماید.

٦-١-٦ تحکیم و یکپارچگی سناریو :

پس ار سناریو تبادل اندیشه، چنانچه سناریوهای مشابه معقول و مستدلی وجود داشته باشد با هم ادغام می شوند. برای انجام این کار facilitator از stakeholder می خواهد تا آن سناریوهایی که در محتوا با هم مشابهند را تعیین کند تا سناریوهای مشابه ادغام شوند. در زمان ادغام سناریوها نظرات و رأی ها متمرکز می شوند و سپس پالایش می شوند.

٦-١-٧ اولویت بندی سناریو :

در این مرحله سناریو ها اولویت بندی می شوند.

٦-١-٨ پالایش سناریو :

پس از مرحله اولویت بندی، بسته به زمان باقی مانده سناریوها پالایش شده و با جزئیات بیشتری بررسی می شود.

نقشه راه (Road map) مربوط به خصوصیات کیفی، در پروسه QAW در زیر آورده شده است. (شکل ۹)

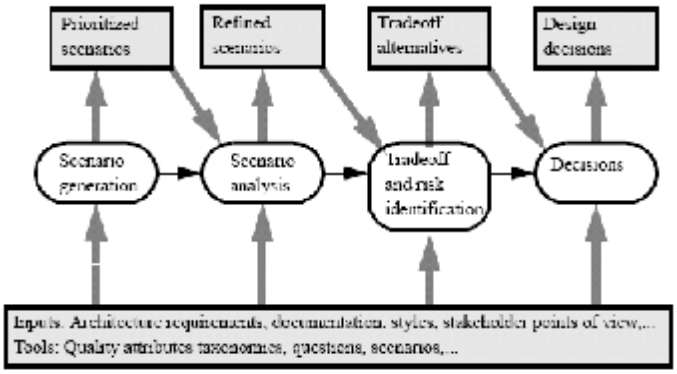


Figure 1: Roadmap Activities

شکل ۹: نقشه راه خصوصیات کیفی در QAW

اطلاعات کسب شده از طریق Quality Attribute WorkShop در موارد زیر قابل استفاده می باشد :

- جهت به روز رسانی دیدگاه معماری
- تسویه و تصحیح نیازمندی های نرم افزار و سیستم
- کمک به راهبرد و توسعه توابع اولیه سیستم
- درک و روشن کردن درایورهای معماری سیستم
- توصیف عملیات سیستم

نتیجه گیری:

یکی از مهمترین چالش های بحث کیفیت نرم افزار به این مورد برمی گشت که استانداری برای آن وجود نداشت. مشاهده شد که درمدل‌های مختلف دیدگاه‌های مختلفی نسبت به خصوصیات کیفی وجود دارد. مزیت استفاده ازمدل های کیفی، سادگی استفاده از آنها و داشتن یک چارچوب و معیارهای برای اندازه گیری خصوصیات کیفی می باشد. درزمان طراحی سیستم بهتر است درمراحل اولیه معماری خصوصیات کیفی شناخته و مصالحه بین آن انجام شود . واگذاری نیازمندی‌های کیفی از مرحله شناخت نیازمندی‌های کیفی به مراحل بعدتر موجب افزایش هزینه (زمانی و پولی) در مراحل بعدی خواهد شد.