۱– معماری نرم افزار چیست و نحوه ارزیابی معماری طرح شده را شرح دهید؟

(راهنمایی : یکی از روشهای کیفی ارزیابی معماری به زبان ساده توضیح دهید)

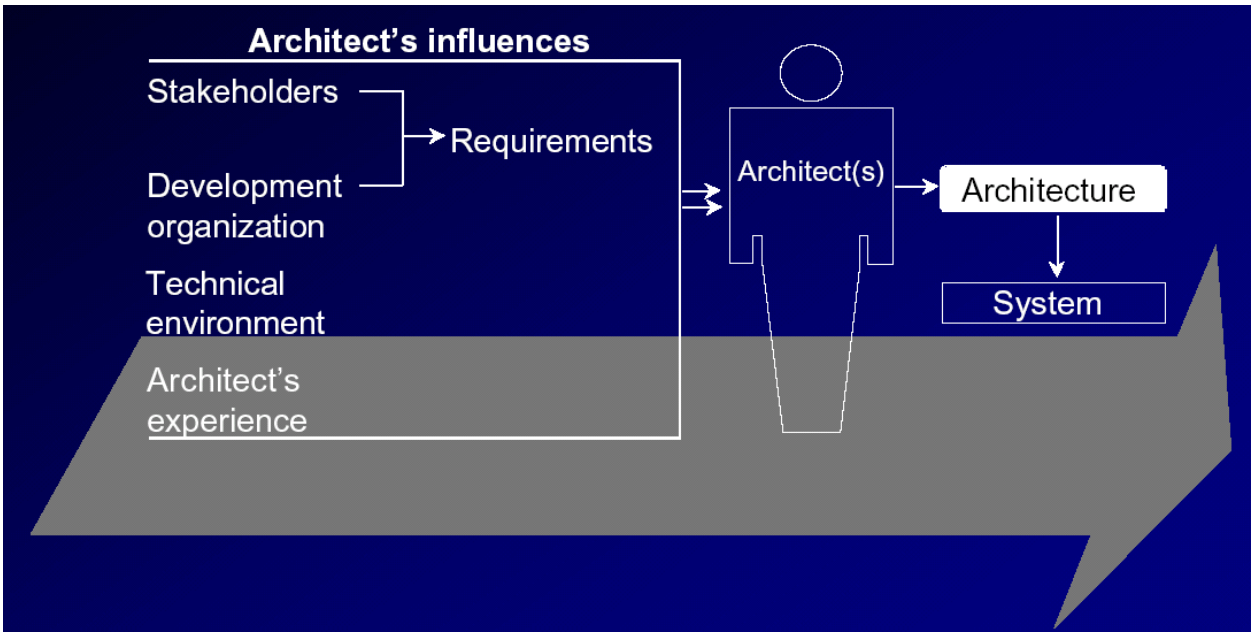
در اکثر پروژه های نرم افزاری موفق، توسعه دهندگان خبره ای که روی آن پروژه کار می کنند، درک مشترکی از طراحی سیستم دارند. این درک مشترک "معماری" نامیده می شود.

معماری یعنی اجزاء یک نرم افزار و رابطه اجزاء با هم

معماری چیست؟

معماری نرم‌افزار یک برنامه یا سیستم محاسباتی، ساختار یا ساختارهای سیستم است که شامل اجزای نرم‌افزار، ویژگی‌های قابل مشاهده خارجی آن اجزا و روابط بین آنها می‌شود

- معماری طراحی سطح بالایی است
- معماری ساختار کلی یک سیستم است
- معماری اجزاء و اتصالات است



معماری بستر تعامل است ، کار معماری تنظیم صفات کیفی(quality attributes) است . معماری پایه و بنیان صفات کیفی نرم افزار است . Quality Attribute

از آنجا که ابزار طراحی کیفیت معماری است، پس تمامی صفات کیفی را با معماری طراحی و پیاده سازی می کنیم و مهندس صفات کیفی را به کمی تبدیل کرده و اندازه می گیرد، پس نیازمندی‌های کیفی و غیر کیفی و عملیاتی با هم طراحی شوند و دیده شوند .

معماری نرم افزار یک سیستم مجموعه ای از ساختارهای مورد نیاز برای استدلال در مورد سیستم است که شامل عناصر نرم افزاری، روابط بین آنها و ویژگی های هر دو است.

معماری مجموعه ای از ساختارهای نرم افزاری است، معماری شامل رفتار است وهمه معماری ها معماری خوبی نیستند.

معماری یک انتزاع است و هر سیستمی یک معماری نرم افزاری دارد

معماری نرم افزار یک سیستم مجموعه ای از ساختارهای مورد نیاز برای استدلال در مورد سیستم است که شامل عناصر نرم افزاری، روابط بین آنها و ویژگی های هر دو است.

یک معمار نرم افزار تصمیمات مهمی در رابطه با نرم افزار می گیرد که در ادامه یکپارچگی کلی آن را مشخص می کند. یک معماری نرم افزار خوب به تعریف ویژگی هایی مانند عملکرد، کیفیت، مقیاس پذیری، قابلیت نگهداری، مدیریت پذیری و قابلیت استفاده کمک می کند.

معماری نرم افزار یک سیستم، تصمیمات طراحی مربوط به ساختار و رفتار کلی سیستم را نشان می دهد. معماری به ذینفعان کمک می کند تا بفهمند و تجزیه و تحلیل کنند که چگونه سیستم به کیفیت های اساسی مانند قابلیت تغییر، در دسترس بودن و امنیت دست می یابد.

معماری نرم‌افزار از تجزیه و تحلیل کیفیت سیستم پشتیبانی می‌کند، زمانی که تیم‌ها در مورد سیستم تصمیم می‌گیرند نه بعد از اجرا، یکپارچه‌سازی یا استقرار. خواه طراحی یک سیستم جدید، تکامل یک سیستم موفق، یا مدرن کردن یک سیستم قدیمی، این تجزیه و تحلیل به موقع تیم ها را قادر می سازد تا تعیین کنند که آیا رویکردهایی که انتخاب کرده‌اند، راه حل قابل قبولی به همراه خواهد داشت یا خیر. یک معماری مؤثر به‌عنوان چسب مفهومی عمل می‌کند که هر مرحله از پروژه را برای همه ذینفعان در کنار هم نگه می‌دارد و چابکی، صرفه‌جویی در زمان و هزینه و شناسایی زودهنگام ریسک‌های طراحی را ممکن می‌سازد.

ایجاد یک معماری موثر که تحویل سریع محصول را برای نیازهای امروزی امکان پذیر می کند و در عین حال به اهداف بلندمدت می پردازد، می تواند چالش برانگیز باشد. ناتوانی در شناسایی، اولویت‌بندی و مدیریت معاوضه‌ها بین کیفیت‌های مهم معماری اغلب منجر به تاخیر در پروژه، دوباره کاری پر هزینه یا بدتر می‌شود.

یک معماری نرم‌افزار مؤثر که توسط شیوه‌های معماری چابک پشتیبانی می‌شود، تکامل مستمر سیستم را امکان‌پذیر می‌سازد. چنین اقداماتی شامل مستندسازی عناصر معماری و روابط متقابل در نظر گرفته شده برای دستیابی به کیفیت های کلیدی است. ارزیابی مکرر معماری برای تناسب اندام با توجه به اهداف تجاری و ماموریت سازمان؛ و تجزیه و تحلیل سیستم مستقر برای انطباق با یک معماری. هنگامی که این شیوه ها به درستی انجام شوند، کیفیت محصول قابل پیش بینی، مشکلات پایین دستی کمتر، صرفه جویی در زمان و هزینه در یکپارچه سازی و آزمایش و تکامل سیستم مقرون به صرفه را امکان پذیر می کنند.

معمار قبل از نوشتن اولین خط کد منبع، ضروری است که به طور روشمند از طریق معماری نرم افزار برای توسعه مؤثر فکر کند.

صفات کیفی یک معماری نرم افزار

- **Usability** قابلیت استفاده - اندازه گیری توانایی کاربر برای استفاده موثر از یک سیستم. Usability یکی از مهمترین ویژگی ها است، زیرا بر خلاف موارد با سایر ویژگی ها، کاربران می‌توانند مستقیماً ببینند که این ویژگی سیستم چقدر خوب کار می کند.
- **Functionality** عملکرد - توانایی سیستم برای انجام کاری که برای آن در نظر گرفته شده است
- **Interoperability** : قابلیت همکاری یک ویژگی از سیستم یا بخشی از سیستم است که وظیفه عملیات آن و انتقال داده ها و تبادل آن با سایر سیستم های خارجی را بر عهده دارد.
- **Modifiability** قابلیت تغییر(اصلاح پذیری) - توانایی ایجاد تغییرات سریع و مقرون به صرفه در یک سیستم . Modifiability تعیین می کند که برای ایجاد تغییرات در هر یک از آیتم های منفرد، چند تغییر مشترک باید در سیستم ایجاد شود.
- **Subsetability** قابلیت زیرمجموعه
- **Reliability** قابلیت اطمینان - توانایی سیستم برای ادامه کار در طول زمان. Reliability ویژگی سیستم مسئول توانایی ادامه کار تحت شرایط از پیش تعریف شده است.
- **Conceptual integrity** یکپارچگی مفهومی – معماری باید کارهای مشابه را به روش های مشابه انجام دهد
- **Performance** عملکرد - پاسخگویی سیستم - پاسخ سیستم را به انجام اقدامات معین برای یک دوره زمانی مشخص نشان می دهد.

- **Availability** در دسترس بودن - نسبت زمانی که سیستم راه اندازی و کار می کند (تأخیر بین خرابی ها و زمان لازم برای از سرگیری عملیات عادی) .Availability بخشی از قابلیت اطمینان است و به صورت نسبت زمان موجود سیستم به کل زمان کار بیان می شود.
- **Testability** آزمون پذیری، تست پذیری نشان می دهد که سیستم تا چه حد امکان انجام تست‌ها را بر اساس معیارهای از پیش تعریف شده فراهم می کند.
- **Security** امنیت ، سیستم مسئول توانایی سیستم در کاهش احتمال اعمال مخرب یا تصادفی و همچنین احتمال سرقت یا از دست دادن اطلاعات است.
- **Maintainability** قابلیت نگهداری توانایی سیستم برای پشتیبانی از تغییرات است.
- **Scalability** مقیاس پذیری توانایی سیستم برای مدیریت افزایش بار بدون کاهش عملکرد یا امکان افزایش سریع بار است.
- **Reusability** قابلیت استفاده مجدد فرصتی برای استفاده از یک جزء یا سیستم در سایر اجزا/سیستم ها با تغییرات کوچک یا بدون تغییر است.
- **Supportability قابلیت** پشتیبانی، توانایی سیستم در ارائه اطلاعات مفید برای شناسایی و حل مشکلات است.

یک معماری نرم افزار سازمان یافته به اطمینان از طول عمر کیفیت داخلی نرم افزار کمک می کند.

هدف معماری نرم افزار به حداقل رساندن نیروی انسانی در تمامی مراحل توسعه، استقرار و نگهداری است.

از ویژگی های معماری نرم افزار خوب می توان به موارد ذیل اشاره نمود :

۱. عملکرد بدون وقفه: سیستم نرم افزار همانطور که در نظر گرفته شده بدون هیچ اشکال یا وقفه ای کار بکند.

۲.Reliability قابلیت اطمینان: سیستم نرم افزار بدون توجه به محیط و تعداد ورودی های داده شده، عملکرد بهینه ای دارد.

۳. قابلیت نگهداری: ایجاد تغییرات نوآورانه در نرم افزار بدون وقفه در عملکرد سیستم موجود کارآمد و ساده است.

۴.Security امنیت: این نرم افزار در برابر انواع حملات اعم از خارجی یا داخلی ایمن است.

۵. حداقل بدهی فنی: کد منبع به جای آشفتگی یا مملو از ضد الگوها، تمیز و منظم است. این امر تلاش‌های بازآفرینی کد و بدهی فنی مربوطه را کاهش می‌دهد.

۶. آزمون پذیری: تست نرم افزار با کیفیت را تضمین می کند زیرا شناسایی سریعتر اشکالات را امکان پذیر می کند. این، به نوبه خود، به اطمینان از زمان سریع‌تر برای بازاریابی نرم‌افزار قابل اجرا کمک می‌کند.

۷. مدولار بودن: با داشتن یک معماری نرم افزاری خوب، تقسیم سیستم نرم افزاری به ماژول های کوچکتر و قابل مدیریت آسان‌تر است. این همچنین در هنگام معرفی معماری میکروسرویس هنگام افزایش مقیاس عملیات کمک می‌کند.

معماری نرم افزار: مجموعه تصمیمات کلان و کلیدی است که برای تولید یک محصول نرم افزاری اتخاذ می‌شود. «عملیات معماری نرم افزار» اثرگذارترین فعالیت بر تضمین کیفیت و مدیریت ریسک و نقطه اوج مهارت و دانش مهندسی نرم افزار در یک پروژه نرم افزاری است. متاسفانه در ایران مهندسین کمی می‌توانند متدلوژی ساختارمندی برای «عملیات معماری» ارائه دهند. تا جایی که مشکلات بسیاری از محصولات ایرانی را می توان در این خلاء ریشه یابی کرد.

با گسترش روز افزون استفاده از مدل‌های فرآیند مبتنی بر معماری، طراحی معماری نرم افزار اهمیت ویژه‌ای یافته است. یک طراحی معماری خوب، طراحی است که نیازهای کیفی مورد انتظار مشتری را برآورده نماید..

معماری نرم افزار یک برنامه یا سیستم کامپیوتری، ساختار یا ساختارهایی از سیستم می باشد، که در برگیرنده اجزاء، صفات قابل مشاهده آن اجزا و ارتباط بین آنها باشد .

از تعریف فوق می توان به نتایج زیر دست یافت:

- معماری، اجزای نرم افزار را تعریف می نماید. همچنین در این تعریف، از جزئیاتی از اجزا، که در نحوه استفاده و ارتباط با اجزای دیگر کاربردی ندارند؛ صرف نظر می گردد .
- هر سیستم نرم افزار شامل چندین ساختار می باشد؛ و هیچ یک از این ساختارها، به تنهایی معماری نرم افزار نمی‌باشد .بلکه این ساختارها در کنار یکدیگر معماری نرم افزار را تشکیل می دهند .
- هر سیستم نرم افزاری دارای یک معماری می باشد. (زیرا هر سیستم نرم افزاری دارای اجزایی است که این اجزا با یکدیگر دارای رابطه می باشند.)
- رفتار هریک از اجزاء، بخشی از معماری نرم افزار می باشد. (زیرا این رفتار در نحوه ارتباط بین اجزا تاثیرگذار است.)
- معماری نرم افزار باید قابل ارزیابی باشد تا بتوان از روی آن تشخیص داد سیستم مورد نظر بر پایه معماری انتخاب شده نیازهای خود را برآورده خواهد کرد یا خیر

معماری خوب نرم افزار به حفظ صفات کیفی نرم افزار در طول عمر آن کمک می کند. همچنین به تقسیم نرم افزار به میکروسرویس ها کمک می کند که مدیریت آسان تر شود. با این حال، هیچ کس نمی تواند آن را در اولین بار درست انجام دهد. حتی بهترین پروژه‌های توسعه محصول نرم‌افزاری نیز دچار مشکل می‌شوند، اما تیمی از متخصصان چابک می‌توانند زودتر آن‌ها را شناسایی و مدیریت کنند. در نهایت، معماری خوب در درازمدت سودآور است زیرا مقیاس‌پذیری و تکامل آن آسان‌تر است. این می تواند در زمان توسعه دهندگان صرفه جویی کند در حالی که نیازهای متغیر مشتریان را برآورده می کند.

سازه‌های معماری به سه دسته تقسیم می‌شوند که نقش مهمی در طراحی، مستندسازی و تحلیل معماری دارند:

ارزشیابی معمولاً به یکی از سه شکل زیر انجام می شود:

- **ارزیابی توسط طراح در فرآیند طراحی**
- **ارزیابی توسط همتایان در فرآیند طراحی**
- **تجزیه و تحلیل توسط افراد خارجی پس از طراحی معماری**

ارزیابی معماری نرم افزار جهت توسعه کیفیت نرم افزار است که میزان تلاش و هزینه های توسعه نرم افزار را کاهش می دهد ، و کیفیت نرم افزار را با اعتبار سنجی نیازمندی کیفی و شناسایی ریسک‌های بالقوه افزایش می‌دهد . با توجه به قابلیت نگهداری ، قابلیت استفاده مجدد و کارایی ، تکنیک‌های گوناگونی برای ارزیابی معماری‌های نرم افزار وجود دارد .

چرا یک معماری را ارزیابی کنیم؟

- هر چه زودتر مشکلی را در پروژه SW پیدا کنید، وضعیت بهتری دارید (هزینه رفع خطا در مرحله طراحی اولیه بسیار کمتر از هزینه رفع همان خطا در اجرا/آزمایش است)
- معماری، اولین نقطه در پروژه است که در آن مبادلات قابل مشاهده است
- معماری ساختار پروژه را تعیین می کند: برنامه ها، بودجه ها، شاخص های عملکرد، ساختار تیم، فعالیت های تست و نگهداری
- مدیریت ریسک

ارزیابی معماری طرح شده تکنیکی است که خصوصیات، نقاط ضعف و قوت معماری نرم افزار و یا سبک و الگوهای طراحی را تعیین می‌کند. ارزیابی معماری در خصوص دستیابی به نیازهای کیفی مطلوب، اهداف کسب و کار و کیفیت معماری است. ارزیابی معماری نرم افزار با انتخاب معماری مناسب این اطمینان را به توسعه دهندگان میدهد که نیازهای وظیفه مندی و غیروظیفه مندی برای کاربران فراهم خواهد شد .درخواست کاربران از یک سیستم نرم افزاری به دو بخش وظیفه مندی و غیر وظیفه مندی تقسیم می شود. **نیازهای وظیفه مندی** به توانایی و عملکرد سیستم در انجام وظایف مربوط می شودکه نیازهای وظیفه‌مندی به دلیل بدیهی بودن در پایان چرخه تولید نرم افزار میزان دستیابی به آنها کاملا واضح و مشخص است و **نیازهای غیروظیفه‌مندی** نیازهای کیفی است از قبیل امنیت ،کارایی، قابلیت اطمینان و... می باشد.

ارزیابی معماری جهت شناسایی نقاط ضعف و قوت در معماری های مختلف در مراحل اولیه توسعه نرم افزار مورد استفاده قرار گیرد، نوعی از روشهای ارزیابی نیز وجود دارد که در سیستم های موجود نیز می‌تواند مفید واقع شود.

- ارزیابی مبتنی بر تجربه: بر پایه تجربه و قلمرو دانش از توسعه دهندگان و یا مشاوران است.افرادی هستند که پیش از اینکه با معماری درگیر شوند می توانند درجه صحت معماری را تعیین کنند.
- ارزیابی مبتنی بر شبیه سازی: ارزیابی مبتنی بر شبیه سازی تکیه بر اجرای تمام یا بخشی از اجزای سازنده در معماری نرم افزار دارد و می تواند برای ارزیابی نیازهای کیفی همانند کارایی استفاده شود.
- مدل سازی ریاضی: مدل سازی ریاضی با استفاده از دلایل ریاضی و متدهایی برای ارزیابی نیازمندی‌های کیفی عملکرد اصلی همانند کارایی و قابلیت اطمینان مورد استفاده قرار گیرد.
- مدل سازی بر پایه سناریو: ارزیابی معماری بر پایه سناریو تلاش می کند تا ارزیابی ویژگی کیفی خاص با ایجاد یک حالت از نیازمندیهای کیفی نمایش دهد.سناریوها شرحی مختصر از تعامل منفرد ذینفع با سیستم می‌باشند.

مزایا و هزینه ها ارزیابی معماری

(+) •توضیح واضح معماری را مجبور می کند

(+) •سهامداران (ذینفعان) را در یک اتاق قرار می دهد

(+) •اهداف متضاد را شناسایی و حل می کند

(+) •شفاف سازی اهداف کیفیت خاص را وادار می کند

(+) •خطرات را در اوایل چرخه زندگی شناسایی می کند

(-) •هزینه (زمان و هزینه)

هر نوع رویکرد سازماندهی شده برای ارزیابی بسیار بهتر از عدم است

روشهای ارزیابی معماری نرم افزار مبتنی بر سناریو

SAAM (Software Architecture Analysis Method)

ATAM (Architecture Tradeoff Analysis Method)

ARID (Active reviews for intermediate designs)

و روشهای دیگر SBAR،ALMA، SAAMCS، ESAAMI،SALUTA، SACAM، DoSAM

SAAM (Software Architecture Analysis Method) روش تجزیه و تحلیل معماری نرم افزار

- بر اساس سناریوها

♣یک سناریو توصیفی از تعامل ذینفعان با سیستم است

سناریوها بسته به دیدگاه هر ذینفع ایجاد می شوند :

- Developer توسعه دهنده - علاقه مند به قابلیت استفاده مجدد پیاده سازی، تعمیر و نگهداری
- Project Manager مدیر پروژه – علاقه مند به زمان، هزینه، کیفیت، توسعه پذیری
- Tester تستر – علاقه مند به قابلیت استفاده، نقشه برداری بر اساس نیازها

مراحل ارزیابی SAAM

شناسایی و جمع آوری سهامداران (ذینفعان)

↓

سناریوها را توسعه و اولویت بندی کنید

↓

توصیف معماری (بررسی واقعی)

↓

سناریوها را به صورت مستقیم یا غیرمستقیم طبقه بندی کنید

↓

ارزیابی سناریو را انجام دهید

↓

تعاملات سناریو را آشکار کنید

↓

ارزیابی کلی ایجاد کنید

سناریوهای SAAM

- سناریوها باید به تکاملی اشاره داشته باشند که سیستم باید از آن پشتیبانی کند (براساس الزامات و نیازمندی‌ها)
  - Functionality عملکرد
  - Development activities فعالیت های توسعه ای
  - Change activities تغییر فعالیت ها
- سناریوها باید وظایف مربوط به همه ذینفعان را نشان دهند
- پیشنهاد: ۱۰-۱۵ سناریو اولویت بندی شده
- سناریوها را می توان در دو کلاس طبقه بندی کرد
  - سناریوهای مستقیم نیازی به تغییرات سیستم ندارند
  - سناریوهای غیر مستقیم نیاز به تغییر سیستم دارند

ارزیابی سناریوی SAAM

- برای هر سناریوی مستقیم، مشاهده کرد که آیا سناریو را می توان با وضعیت فعلی سیستم انجام داد
- برای هر سناریوی غیر مستقیم
  - اجزایی را که باید اصلاح، اضافه یا حذف شوند، شناسایی نمایید.
  - تخمین دشواری اصلاح (بر اساس تعداد مؤلفه هایی که باید اصلاح شوند و تأثیر اصلاح)

تعامل سناریوی SAAM

- سناریوهای غیرمستقیم متعددی که بر یک مؤلفه تأثیر می‌گذارند می‌تواند نشان‌دهنده یک مشکل باشد
  - Could be good می تواند خوب باشد: اگر سناریوها انواع دیگری از یکدیگر باشند
  - Could be bad ممکن است بد باشد: نشان دهنده تفکیک ضعیف مسئولیت هاست

نتایج ارزیابی SAAM

- طبقه بندی سناریوها بر اساس اهمیت
- تصمیم گیری در مورد پذیرش یا اصلاح معماری

نتیجه گیری هر نوع رویکرد سازمان یافته برای ارزیابی بسیار بهتر از هیچ کدام است.

سؤالات نیمه دوم

۱- مدل یک صفت کیفی را بطور کامل شرح دهید.

از آنجایی که معماری پایه و بنیان صفات کیفی نرم افزار است که همان Quality Attribute ها هستند که آنها را باسناریو ها تعریف می کنیم و با history ها آنها را بیان می کنیم ، و با تاکتیک ها و تکنیکهای خاصی آنها را اعمال می کنیم بستگی به کیفیتی که مد نظر باشد .

از آنجایی که نرم افزار اهمیت دارد ، معماری نیز اهمیت دارد، سهم صفات کیفی است که چگونه ایجاد می‌شوند .صفات کیفی با چیدمان اجزا و روابط تعریف می شوند . و اگر سیستم اجتماعی را بخواهیم اصلاح کنیم ، باید صفت کیفی را تغییر دهیم . به عبارتی نگاه نرم افزاری ، صفت کیفی و چیدمانی که به می‌تواند کمک کند تا خیلی از مسائل اجتماعی را درک کنیم و بعد امید به ما می دهد که می‌توانیم، یعنی بتوانیم ، آزمون خطا نیست ،باید تاکتیک داشته باشیم ، بایستی الگو داشته باشیم ، بایستی بدانیم که چه کار باید بکنیم تا سیستم درست بشود .دستوری نیست ، بایستی هزینه پرداخت کنیم

و از آنجایی که منظور مهندسی کمیت است و آن هم شامل اندازه ،نسب و تناسب می باشد.و ما باید کیفیت را به کمیت تبدیل کنیم ، یعنی بتوانیم اندازه گیری کنیم ، با مقیاس مشخص (متر)،

پس ما بایستی بفهمیم مکانیسم تبدیل کیفیت به کمیت چیست ؟ ما چطوری صفت کیفی(Quality Attribute) را می توانیم اندازه گیری کنیم ؟ می توانیم تشخیص دهیم ؟ و در ابتدا تعریف کنیم ؟

ما *چه مطالبی را در رابطه با صفات کیفی در نظر بگیریم ، نیازمندیهای کیفی را چگونه بیان می کنیم ، چگونه یک صفت کیفی مورد نظر را پیاده سازی کنیم ، چگونه تصمیم سازی های خود را کلاسیک کنیم ؟ الگوریتمی کنیم ؟ ساختارمند کنیم ؟ تا به یک صفت کیفی برسیم .*

یک نیازمندی صفت کیفی باید بدون ابهام و قابل آزمایش باشد. ما از یک فرم مشترک برای تعیین تمام نیازمندیهای صفات کیفی استفاده می کنیم. این مزیت تاکید بر مشترکات بین تمام صفات کیفی است.

پس جهت تعریف صفت کیفی ، یک فرم مشترک و عمومی را باید تعیین بکنیم و تمام نیازمندیهای صفات کیفی را در آن فرم و قالب تعریف کنیم که به آن سناریو می گوییم .

بیان صفات کیفی در قالب سناریوها داری ۶ بخش است که برای هر صفت کیفی این ۶ قسمت را باید تعریف کرد

۱) **Stimulus** محرک
۲) **Stimulus source** منبع محرک
۳) **Response** واکنش
۴) **Response measure** اندازه گیری واکنش
۵) **Environment** محیط
۶) **Artifact** غیرواقعی، ساختگی

یعنی برای هر صفت کیفی سناریویی تعریف کردند که در حالت استاندارد داری شش بخش فوق می باشد و اگر شما می خواهید یک صفت کیفی تعریف کنید بایستی برای آن شش بخش رامعین کنید ،

Stimulus source منبع محرک (منبع تحریک): یک محرک باید منبعی داشته باشد - باید از جایی آمده باشد.منبع محرک ممکن است بر نحوه برخورد سیستم با آن تأثیر بگذارد.درخواست یک کاربر قابل اعتماد مانند درخواست یک کاربر نامعتبر مورد بررسی دقیق قرار نخواهد گرفت.

منبع تحریک موجودی است که محرک را ایجاد کرده است(یک انسان، یک سیستم کامپیوتری یا هر محرک دیگری).صفت کیفی کی بکار می آید؟ کی مد نظر قرار می گیرد ؟کی مورد توجه قرار می گیرد؟

Stimulus محرک. محرک شرایطی است که وقتی به یک سیستم می رسد نیاز به پاسخ دارد. (*آن شرایطی است که سیتم باید به آن شرایط پاسخ دهد*) . ما از اصطلاح "محرک" برای توصیف رویدادی که به سیستم می رسد استفاده می کنیم. محرک می تواند یک رویداد برای جامعه عملکرد، یک عملیات کاربر برای جامعه قابلیت استفاده، یا یک حمله به جامعه امنیتی باشد. ما از محرک برای توصیف یک اقدام انگیزشی برای کیفیت های

رشدی استفاده می کنیم. بنابراین، یک محرک برای تغییرپذیری، درخواستی برای اصلاح است. یک محرک برای آزمایش پذیری، تکمیل مرحله توسعه است.

Environment محیط. محرک تحت شرایط خاصی رخ می دهد. سیستم ممکن است در وضعیت اضافه بار یا درحالت عادی کار کند یا دربرخی وضعیت‌های مرتبط دیگر باشد. برای بسیاری از سیستم‌ها، عملکرد "عادی" می تواند به یکی از تعدادی از حالت ها اشاره کند. (*آن شرایطی که رخداد* (**Stimulus**) *به سیستم وارد می شود*)

Artifact غیرواقعی، ساختگی(*کدام بخش از سیستم*). (*بخشی از سیستم که به آن* **Stimulus** *باید پاسخگو باشد*) برخی از مصنوعات تحریک می شود. این ممکن است مجموعه‌ای از سیستم‌ها، کل سیستم یا بخشی از آن باشد.

Response واکنش(پاسخش چیست ، باید به سیستم تعریف شود وقتی **Stimulus** *آمد چه کار باید بکنیم* ،در شرایط مختلف چه کار باید کرد). پاسخ فعالیتی است که در نتیجه ورود محرک انجام می شود.

Response measure اندازه گیری واکنش(**Response measure** *مهمترین قسمت است که کیفیت را به کمیت تبدیل کنیم ،در این قسمت است که ما می توانیم اندازه گیری کنیم*) هنگامی که پاسخ رخ می دهد، باید به نوعی قابل اندازه گیری باشد تا بتوان این نیاز را آزمایش کرد.(*وقتی سیستم به من پاسخ داد چگونه آن را اندازه بگیرم . وقتی modifiable است یا نیست ، شما تبدیل کن به زمان و هزینه یعنی هزینه را اندازه بگیر وزمان را اندازه بگیر یعنی اگر در آن محدوده زمانی بود modifiable است و اگر نبود modifiable نیست ، وقتی می گوید* **Availability باید نگاه کرد چند درصد زمان سیستم کار می کند . وقتی می گوید Performance است باید نگاه کرد کی Stimulus** *دادیم و کی* **Response** **آمد یعنی تفاوت زمانی را بگیر یعنی زمان دلتا t محاسبه کن و**

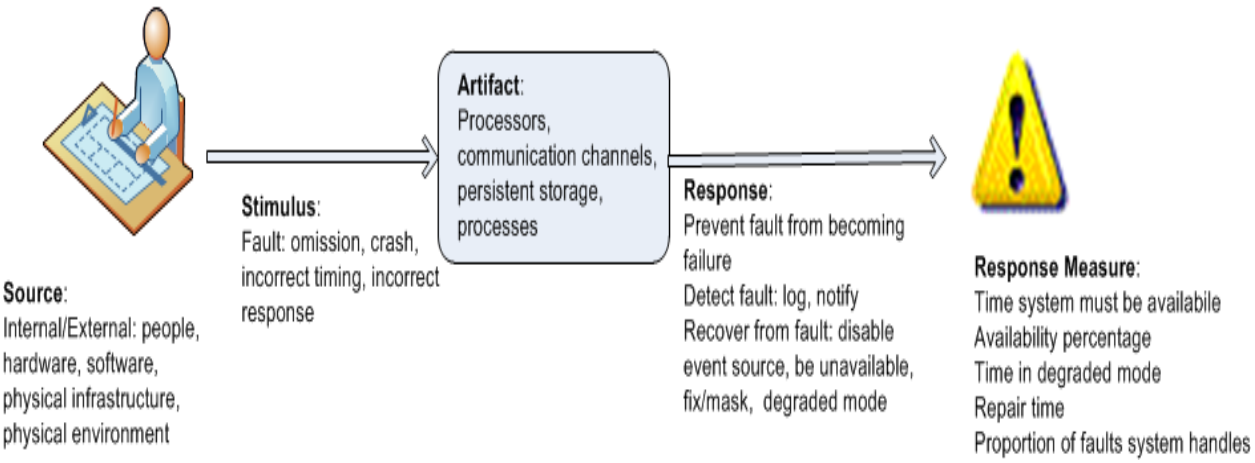
اگر در محدوده خاص بود Performance دارد و اگر نبود بنابراین Performance ندارد . به همین ترتیب صفات کیفی دیگر)

بستگی به **quality attributes** که شما دارید این شش مورد فوق است که باید تعریف شوند ، این ها در قالب سناریو ها تعریف می‌شوند و دو نوع سناریو تعریف می شود **scenarios general** و **concrete scenarios**

scenarios general :شکل کلی و قالب کلی یا چارچوب کلی تعریف سناریو ها هستند یعنی موارد صفات کیفی را باید در یک قالب تعریف کنیم یعنی ۶ جزء برای آنها بیان کنیم

concrete scenarios یعنی آنچه که شما می خواهید یعنی در قالبی که تعریف شده شما چه می‌خواهید

نمونه ای از سناریوی کلی (general scenarios)برای صفت کیفی در دسترس بودن(availability):



صفات کیفی (quality attributes) با تاکتیک ها پیاده سازی می شوند ، تاکتیک همان تصمیم مهندسی است ، تاکتیک ها پایه طراحی معماری هستند . تاکتیک ها تجربه های مفید و کارهای مثبتی هستند

بنابراین جهت دستیابی به صفات کیفی از طریق تاکتیکها اقدام می کنیم .تاکتیکها مجموعه‌ای از تکنیک‌های طراحی اولیه (primitive design)هستند که یک معمار می‌تواند برای دستیابی به پاسخ صفات کیفی از آنها استفاده کند. که به‌این تاکتیک های اولیه طراحی معماری می‌گوییم. تاکتیک‌ها، مانند الگوهای طراحی (design patterns) (تاکتیک هایی که همبند هستند)، تکنیک هایی هستند که معماران سال‌ها از آنها استفاده می‌کنند. ما تاکتیک‌ها را اختراع نمی‌کنیم، ما به سادگی آنچه را که معماران درعمل انجام می‌دهند، به تصویر می‌کشیم.

از آنجایی که نرم افزار معماری دارد و برای آنکه معماری متناسب باشد با آن صفت کیفی، و ما را به آن صفت کیفی هدایت بکنه ، لازم است که ما یک سری تاکتیکها را استفاده کنیم و با توجه به اینکه تاکتیکها تصمیمات مهندسی هستند و کمک می کنند تا تصمیم بگیریم چه کار کنیم

و اگر فرد بتواند موارد ذیل را رعایت کند یعنی تاکتیکی را بکارگیری بکند و تعریف بکند نیاز است یک چک لیستی تهیه شود

ما تصمیمات طراحی را که یک معمار باید اتخاذ کند به شرح زیر دسته بندی می کنیم:

۱) تخصیص مسئولیت ها
۲) مدل هماهنگی
۳) مدل داده
۴) مدیریت منابع
۵) نقشه برداری در بین عناصر معماری
۶) تصمیمات زمانی الزام آور
۷) انتخاب تکنولوژی و فناوری

با در نظر گرفتن موارد فوق در چک لسیت و بررسی آنها می شود صفت کیفی متناسبی را پیاده سازی نمود.

۲– یک صفت کیفی تخیلی را طراحی و جدول سناریوهای عمومی آن و یک نمونه سناریوی معین برای آن تعریف نمائید .

ازآنجایی که من نه تحریرگر خوب و نه رویا پرداز خوب هستم ، اما با عنایت به اینکه همه این روایتها و تخیلات که ذهن انسان نقش می بندد ، بیشتر بر اساس واقعیت می باشد و ما انسانها از بعضی امور اطراف خود که وجود خارجی دارند غافل می‌باشیم. برای مثال تولد ، زندگی ، تنفس ، سلول ، ویروس و ... . تا زمانی که با یکی از اعضای خود احساس نکنیم وجود آنها را درک نمی کنیم و به آنها هم فکر نمی کنیم

صفتی که من در نظر گرفتم صفت **زنده** است، افراد به علت عواملی مانند بیماری ،حوادث طبیعی و غیر طبیعی و ... مورد فراز و نشیب زندگی می شوند .امیدوارم بتوانم گوشه ای از دانش آموخته خود از استاد گرامی را به تحریر بیاورم .

Alive_ability General Scenario

Portion of Scenario	Possible Values
Source	Internal/external: People, hardware, physical infrastructure, physical environment
Stimulus	Fault : omission, crash, incorrect timing, incorrect response, Sickness
Artifact	System’s processors, communication channels, persistent storage, processes
Environment	Normal operation, startup, shutdown, repair mode, degraded operation, overloaded operation
Response	Prevent the fault from becoming a failure Detect the fault: • log the fault • notify appropriate entities (people or systems) Recover from the fault • disable source of events causing the fault • be temporarily unavailable while repair is being effected • fix or mask the fault/failure or contain the damage it causes • operate in a degraded mode while repair is being effected
Response Measure	Time or time interval when the system must be available Availability percentage (e.g. ۹۹.۹۹۹٪) Time to detect the fault Time to repair the fault Time or time interval in which system can be in degraded mode Proportion (e.g., ۹۹٪) or rate (e.g., up to ۱۰۰ per second) of a certain class of faults that the system prevents, or handles without failing

Portion of Scenario	Possible Values
Source	مردم، سخت افزار(خودرو ، موتور ، لوازم محل کار و ...)،بیماری ، زیرساخت‌های موجود (جاده‌ای ،ساختمانی و ... نا امن) ، محیط زندگی (گرما، سرما ، دریا، جنگل ،کوه و ...)
Stimulus	مرگ، تصادف، کم شدن عمر انسان ، عدم وجود سلامت می شود
Artifact	مغز، قلب و عروق ، حافظه و سایر اندام ها
Environment	شرایط زندگی نرمال و عادی ،بیداری، خواب ، معالجه و درمان ، عملکرد عضوهای صدمه دیده ، عملکرد شخص وقتی فشار زندگی زیاد است
Response	از تبدیل شدن عیب به شکست جلوگیری کنید تشخیص عیب: عیب را ثبت کنید به نهادهای مناسب (افراد (پزشکان و ....)یا بیمارستانها ها و مراکز درمانی و ... ،) اطلاع دهید از عیب خلاص شوید عاملی که باعث بروز مشکل شده را تشخیص داده و آن را اصلاح کرده و یا از بین ببرید در حین انجام معالجه و درمان موقتاً استراحت کند خطا/شکست را برطرف یا آسیبی که ایجاد می کند را جبران کنید در حالی که مشکلی ایجاد شد و شخص مورد معالجه و درمان قرار گرفت ، بتوان در حالت ناقص بودن و معیوب بودن شخص بتوان کار کند
Response Measure	زمان یا فاصله زمانی که هر شخص باید در حال زیست باشد درصد زیستن (به عنوان مثال ۹۹.۹۹۹٪) زمان تشخیص بیماری و مشکل است زمان معالجه و درمان عیب است زمان یا فاصله زمانی که در آن شخص می تواند در حالت بیماری قرار گیرد نسبت (به عنوان مثال، ۹۹٪) یا نرخ (به عنوان مثال، تا ۱۰۰ در ثانیه) از یک کلاس معین از خطاهایی که سیستم از آنها جلوگیری می کند، یا بدون خرابی آنها را مدیریت می کند.

Sample Concrete Alive_ailability Scenario

The memory and other human organs damaged by the accident and the function of the injured limb are recorded by the emergency officer and notified to the appropriate institutions (doctor, hospital and related medical centers) so that at the time or season when the person may be ill. To be determined

حافظه و سایر اندامهای انسان در اثر تصادف آسیب دیده و عملکرد عضو صدمه دیده توسط مامور اورژانس ثبت و به نهادهای مناسب(پزشک ، بیمارستان و مراکز درمانی مرتبط) اعلام می گردد تا در زمان یا فصله زمانی که آن شخص می تواند در حالت بیماری باشد مشخص شود

با آرزوی سلامتی و شادکامی