کیفیت در مهندسی نرم‌افزار

فهرست مطالب

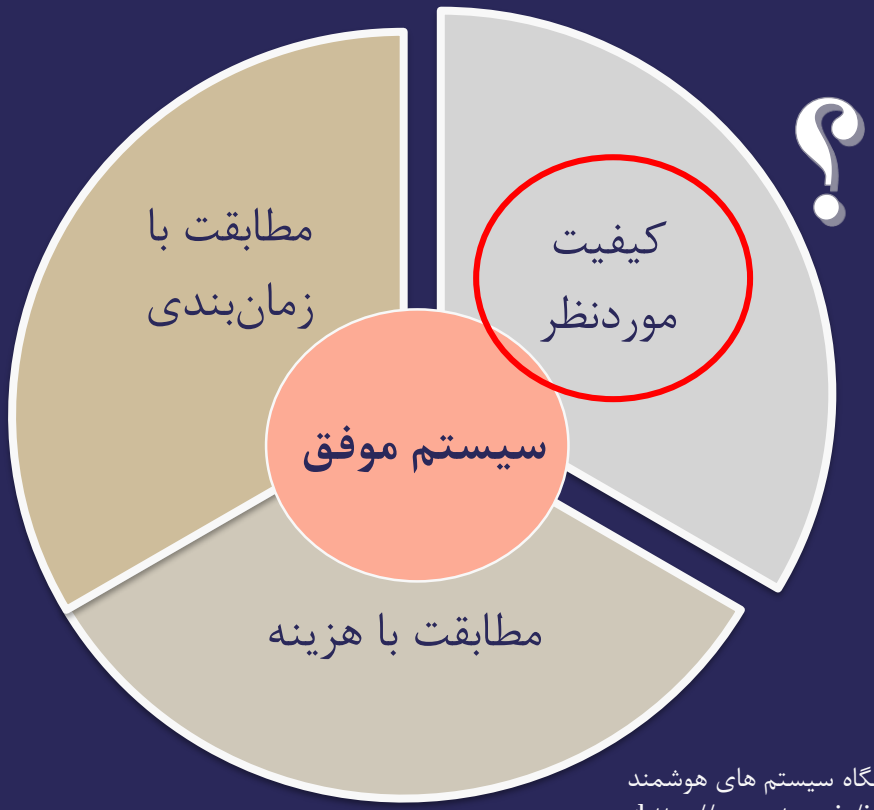
- کیفیت چیست؟
- تعریف کیفیت
- ابعاد کیفیت
- هزینه‌ی کیفیت

کیفیت چیست؟

- کیفیت نرم‌افزار مفهومی است که در ابتدا نمی‌توان به سادگی آن را تعریف نمود. اما مشکلات ناشی از عدم کیفیت نرم‌افزار به راحتی قابل درک هستند. برای مثال:
  - هدر رفتن زمان موثر کاری و صرف شدن آن به انجام دوباره کاری (rework)
  - از دست رفتن داده‌ها
  - عدم رضایت مشتری
  - از دست رفتن فرصت‌ها در شرایط رقابت
  - هزینه های نگه داری بالا
  - down time های مکرر
  - و ...

کیفیت چیست؟

- بنابراین می‌توان گفت کیفیت چیزی است که در صورت عدم برآورده‌سازی آن یک پروژه نرم افزاری با شکست روبرو میشود.



کیفیت چیست؟

- بیان کیفیت از دیدگاه عملی (Pragmatic View)
  - کیفیت از دیدگاه عوامل مختلف در پروژه متفاوت است:
    - دید شهودی: کیفیت چیزی است که می توان آن را فورا تشخیص داد.
    - دید کاربر: اگر سیستم اهداف کاربر نهایی را برآورده کند دارای کیفیت است.
    - دید سازنده: اگر سیستم با مشخصات خود مطابقت کند دارای کیفیت است.
    - دید محصول: کیفیت بر مبنای ویژگی‌ها و کارکردهایی که یک محصول فراهم می‌آورد قابل تعیین است.
    - دید مبتنی بر ارزش: کیفیت را بر مبنای میزان تمایل مشتری برای خرید و پرداخت محصول می سنجد.

در حقیقت کیفیت تمام این دیدگاه ها را در بر می گیرد و حتی می‌تواند دیدگاه هایی دیگری را نیز شامل شود.

تعریف کیفیت

- کیفیت نرم‌افزار

- به کارگیری یک فرآیند نرم افزاری موثر که در اثر آن یک محصول مفید تولید می شود که ارزش های قابل اندازه گیری برای تولید کننده و استفاده کننده آن فراهم می آورد.

تعریف کیفیت

- فرآیند نرم‌افزاری موثر

- یک فرآیند نرم‌افزاری موثر، زیرساختارهای لازم برای تولید یک محصول با کیفیت را فراهم آورده و از آن پشتیبانی می‌کند:
  - مدیریت پروژه (در صورتی که پروژه به درستی مدیریت نشود، بی نظمی و بی برنامگی‌ها به طور قطعی در کیفیت تاثیرگذار خواهند بود)
  - گام‌های تولید محصول: تحلیل- طراحی- پیاده‌سازی- تست- نگه‌داری (در صورتی که هر یک از گام‌ها به درستی انجام نشود کیفیت محصول نهایی به طور قطعی تحت تاثیر قرار می‌گیرد)
  - فعالیت‌های همیشگی مانند مدیریت، مدیریت ریسک

تعریف کیفیت

- محصول مفید
  - محصولی که کارکردها، ویژگی‌ها را به صورتی reliable و بدون مشکل ارائه می‌کند.
  - یک محصول مفید نیازمندی‌های ذی‌نفعان را برآورده می‌سازد.

(نیازمندی های explicit و implicit)

تعریف کیفیت

- ارزش‌های قابل اندازه‌گیری برای تولید کننده و استفاده کننده
  - برای تولید کننده
    - هزینه نگهداری کمتر
    - برطرف سازی اشکالات کم تر
    - ارائه خدمات کمتر پشتیبانی
  - برای استفاده کننده

ابعاد کیفیت Garvin

- Performance Quality
  - آیا نرم‌افزار تمام کارکردها، ویژگی‌ها و محتوای مشخص شده در نیازمندی‌ها را به نحوی که مقادیر قابل اندازه‌گیری برای کاربر فراهم می‌آورد؟
- Feature quality
  - آیا نرم‌افزار ویژگی‌هایی فراهم می‌آورد که کاربر را در اولین استفاده شگفت زده کند؟
- Reliability
  - آیا نرم‌افزار تمام قابلیت‌ها و ویژگی‌های مورد نظر را بدون شکست ارائه می‌کند؟ آیا در مواقع نیاز در دسترس است؟
- Conformance
  - آیا نرم افزار با استانداردهای مرتبط در داخل و خارج از سازمان مطابقت دارد؟ آیا با قراردادهای کدنویسی و طراحی موجود در سازمان هم خوانی دارد؟

ابعاد کیفیت Garvin

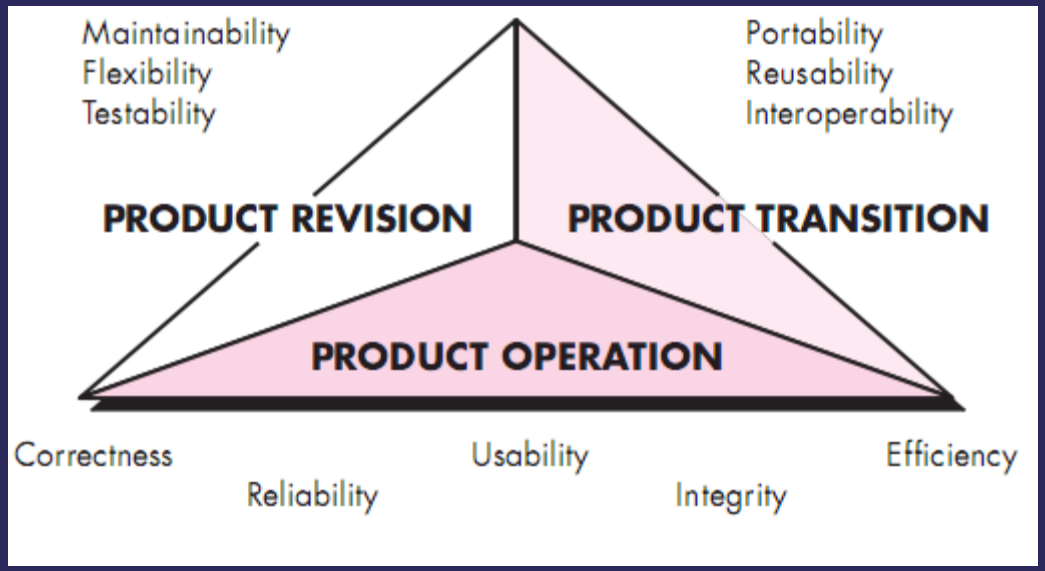
- Durability
  - آیا نگهداری از محصول (اعمال تغییرات و تصحیحات) موجب تاثیرات منفی و نامطلوب در سایر بخش‌ها می‌شود؟
- Serviceability
  - آیا نگهداری از محصول (اعمال تغییرات و تصحیحات) در مدت زمان مورد نیاز قابل اعمال هستند؟
- Aesthetics
  - جنبه‌های زیبایی شناسی محصول
- Perception
  - اعتبار سازنده محصول، می تواند در رابطه با اطمینان به کیفیت محصول تاثیر گذار باشد.

ابعاد کیفیت Garvin

- بسیاری از این ابعاد معرفی شده تنها به صورت توصیفی مطرح شده‌اند و امکان اندازه‌گیری آن‌ها وجود ندارد. بنابراین نمی‌توان کیفیت محصول را بر مبنای به دقت سنجید.

راه حل
به کارگیری سیستم اندازه‌گیری
Metric, measure , Measurement
در رابطه با کیفیت

فاکتورهای کیفیت McCall



فاکتورهای کیفیت McCall

- Product operation

فاکتورهایی کیفی که بیانگر ویژگی های عملیاتی محصول می‌باشند.

- Correctness: میزان درجه‌ای که یک سیستم با مشخصات خود مطابقت داشته و نیازمندی‌های کاربر را برآورده میسازد.
- Reliability: میزان درجه‌ای که یک سیستم کارکردهای مورد نظر خود را با دقت مورد نیاز فراهم می‌آورد. (البته این تعریف دقیقی نیست!)
- Efficiency: میزان منابع محاسباتی مورد نیاز سیستم برای انجام یک کارکرد.
- Integrity: میزان درجه‌ای که دسترسی به نرم افزار یا داده‌ها توسط افراد غیرمجاز، قابل کنترل است.
- Usability: میزان تلاش مورد نیاز برای یادگیری، فهم، فراهم آوردن ورودی‌ها و استفاده عملیاتی کارکردهای سیستم.

فاکتورهای کیفیت McCall

- Product revision

فاکتورهایی کیفی که بیانگر میزان تغییرپذیری محصول می‌باشند.

- Maintainability: میزان تلاش مورد نیاز برای محل‌یابی و از بین بردن خطا.
- Flexibility: میزان تلاش مورد نیاز برای تغییر یک برنامه‌ی عملیاتی
- Testability: میزان تلاش مورد نیاز برای تست یک برنامه جهت اطمینان از مطابقت با کارکردهای مورد نظر.

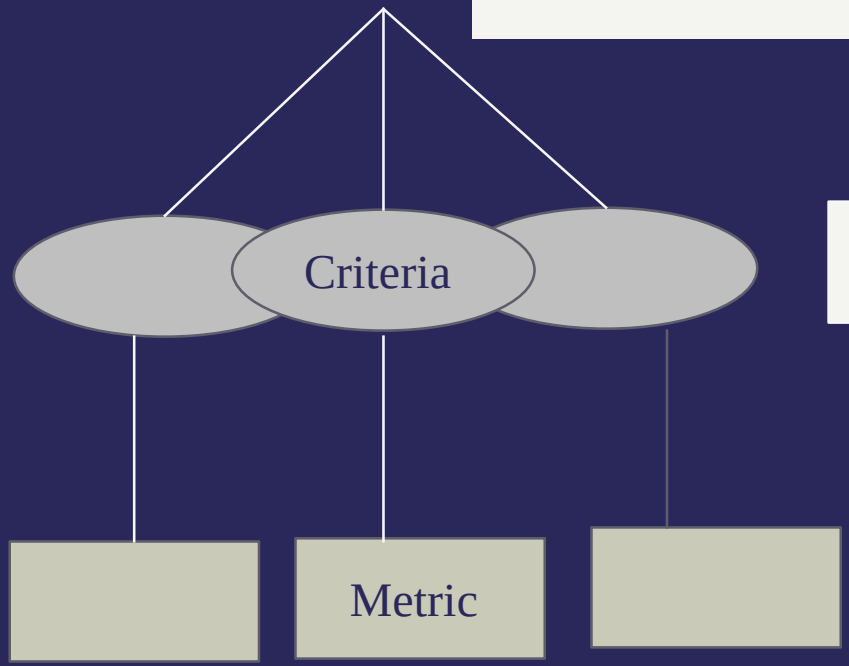
فاکتورهای کیفیت McCall

- Product transition

فاکتورهایی کیفی که بیانگر قابلیت تطبیق‌پذیری محصول با محیط جدید می‌باشند.

- Portability: میزان تلاش مورد نیاز برای انتقال یک برنامه از یک پیکربندی سخت افزاری یا محیط نرم افزاری به پیکربندی یا محیط دیگر.
- Reusability: میزان قابلیت استفاده از سیستم در برنامه های کاربردی دیگر
- Interoperability: میزان تلاش مورد نیاز برای برقراری ارتباط بین سیستم با سیستم های دیگر

فاکتورهای کیفیت McCall
Factor- Criteria- Metric



فاکتورهای کیفیت در بالاترین سطح تعریف می‌شوند و برای اینکه مدیران کسب‌وکار بتوانند اهداف کیفی را بیان کنند مورد استفاده قرار می‌گیرند.

فاکتورهای کیفیت سپس به یک سری معیار شکسته می‌شوند که در رابطه با خصوصیات سطح پایین تر نرم افزار هستند.

در نهایت معیارهای تعریف شده به یک سری متریک شکسته می‌شوند که قابل اندازه‌گیری هستند.

فاکتورهای کیفیت McCall
Factor- Criteria- Metric

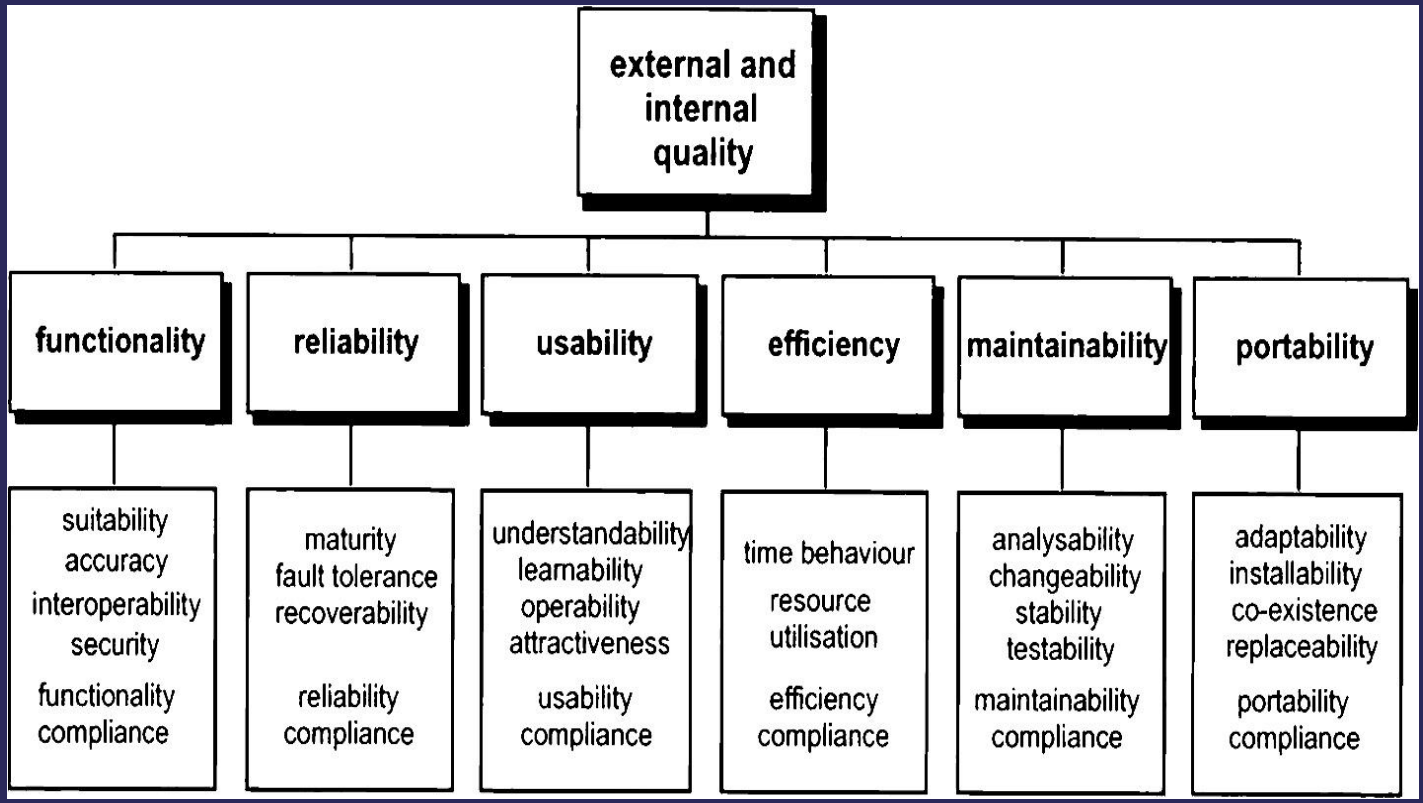
❖ به این ترتیب هر فاکتور به صورت زیر مطرح می‌گردد:

$$f_q = c_1 * m_1 + c_2 * m_2 + \cdots + c_n * m_n$$

فاکتور کیفیت — ضریب اهمیت — متریک

Quality factor \ Software quality metric	Correctness	Reliability	Efficiency	Integrity	Maintainability	Flexibility	Testability	Portability	Reusability	Interoperability	Usability
Auditability				x			x				
Accuracy		x									
Communication commonality										x	
Completeness	x										
Complexity		x				x	x				
Concision			x		x	x					
Consistency	x	x			x	x					
Data commonality										x	
Error tolerance		x									
Execution efficiency			x								
Expandability						x					
Generality						x		x	x	x	
Hardware Indep.								x	x		
Instrumentation				x	x		x				
Modularity		x			x	x	x	x	x	x	
Operability			x								x
Security				x							
Self-documentation					x	x	x	x	x		
Simplicity		x			x	x	x				
System Indep.								x	x		
Traceability	x										
Training											x

ISO 9126


external and internal quality
functionality
reliability
usability
efficiency
maintainability
portability
suitability
accuracy
interoperability
security
functionality compliance
maturity
fault tolerance
recoverability
reliability compliance
understandability
learnability
operability
attractiveness
usability compliance
time behaviour
resource utilisation
efficiency compliance
analysability
changeability
stability
testability
maintainability compliance
adaptability
installability
co-existence
replaceability
portability compliance

Targeted Quality Factors

- **Intuitiveness. The degree to which the interface follows expected usage patterns so that even a novice can use it without significant training.**
  - Is the interface layout conducive to easy understanding?
  - Are interface operations easy to locate and initiate?
  - Does the interface use a recognizable metaphor?
  - Is input specified to economize key strokes or mouse clicks?
  - Does the interface follow the three golden rules? (Chapter 15)
  - Do aesthetics aid in understanding and usage?

Targeted Quality Factors

- **Efficiency. The degree to which operations and information can be located or initiated.**
  - Does the interface layout and style allow a user to locate operations and information efficiently?
  - Can a sequence of operations (or data input) be performed with an economy of motion?
  - Are output data or content presented so that it is understood immediately?
  - Have hierarchical operations been organized in a way that minimizes the depth to which a user must navigate to get something done?

Targeted Quality Factors

- **Robustness. The degree to which the software handles bad input data or inappropriate user interaction.**
  - Will the software recognize the error if data values are at or just outside prescribed input boundaries? More importantly, will the software continue to operate without failure or degradation?
  - Will the interface recognize common cognitive or manipulative mistakes and explicitly guide the user back on the right track?
  - Does the interface provide useful diagnosis and guidance when an error condition (associated with software functionality) is uncovered?

Targeted Quality Factors

- **Richness. The degree to which the interface provides a rich feature set.**
  - Can the interface be customized to the specific needs of a user?
  - Does the interface provide a macro capability that enables a user to identify a sequence of common operations with a single action or command?

The Software Quality Dilemma

- If you produce a software system that has terrible quality, you lose because no one will want to buy it.
- If on the other hand you spend infinite time, extremely large effort, and huge sums of money to build the absolutely perfect piece of software, then it's going to take so long to complete and it will be so expensive to produce that you'll be out of business anyway.
- Either you missed the market window, or you simply exhausted all your resources.
- So people in industry try to get to that magical middle ground where the product is good enough not to be rejected right away, such as during evaluation, but also not the object of so much perfectionism and so much work that it would take too long or cost too much to complete.

“Good Enough” Software

- **Good enough software delivers high quality functions and features that end-users desire, but at the same time it delivers other more obscure or specialized functions and features that contain known bugs.**
- Arguments *against* “good enough.”
  - It is true that “good enough” may work in some application domains and for a few major software companies. After all, if a company has a large marketing budget and can convince enough people to buy version 1.0, it has succeeded in locking them in.
  - If you work for a small company be wary of this philosophy. If you deliver a “good enough” (buggy) product, you risk permanent damage to your company’s reputation.
  - You may never get a chance to deliver version 2.0 because bad buzz may cause your sales to plummet and your company to fold.
  - If you work in certain application domains (e.g., real time embedded software, application software that is integrated with hardware can be negligent and open your company to expensive litigation.

هزینه کیفیت
the cost of quality

- هزینه‌های جلوگیری
  - هزینه هایی هستند که به صورت کلی برای برنامه‌ریزی و هماهنگی تمام برنامه‌های کنترل و تضمین کیفیت موردنیاز هستند، و به پروژه یا یک سیستم به خصوص مرتبط نیستند. این هزینه‌ها عبارتند از:
    - چک‌لیست‌ها، تمپلیت‌ها
    - متریک‌های کیفیت نرم‌افزار
    - ابزارهای مدیریت پیکربندی
    - رویه‌ها و دستورالعمل‌ها
    - هزینه‌های آموزش
    - ابزارهای مورد نیاز جهت تست

هزینه کیفیت
the cost of quality

- هزینه‌های برآورد

- هزینه‌هایی هستند که مربوط به فعالیت‌های می‌شوند که به منظور شناسایی error و defect در یک پروژه به خصوص مورد استفاده قرار می‌گیرند:
  - هزینه مرور
  - هزینه تست و اشکال زدایی

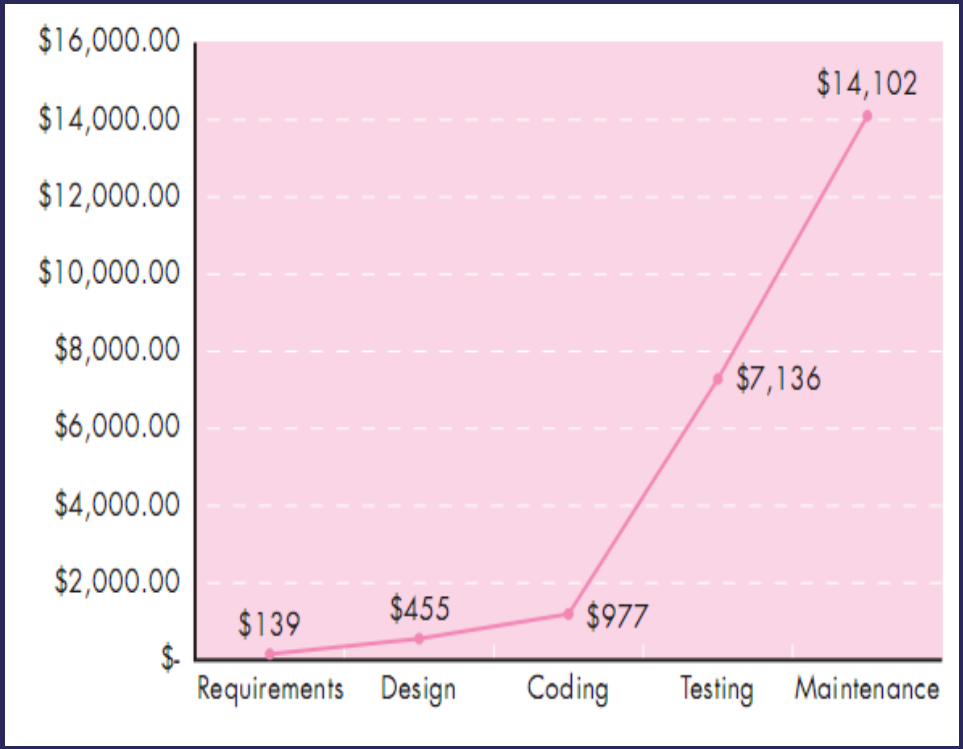
هزینه کیفیت
the cost of quality

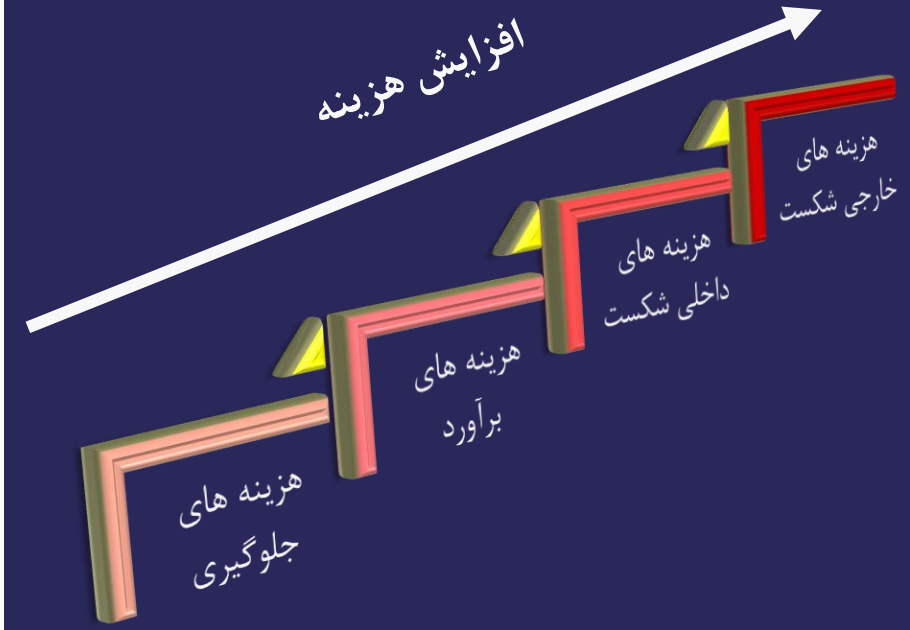
❖ هزینه‌های شکست

هزینه هایی هستند که به علت شکست در جلوگیری و برطرف سازی خطا ایجاد می شوند و بسته به این که قبل از تحویل یا بعد به مشتری شناسایی شوند به دو دسته ی زیر تقسیم می‌شوند:

- هزینه های internal
  - هزینه های انجام rework برای تصحیح error و defect
  - هزینه های تست رگرسیون و یا بررسی مجدد در اثر تاثیرات منفی تصحیحات انجام شده در rework
- هزینه های external
  - هزینه های برطرف سازی شکایات مشتریان
  - هزینه های بازگشت و جایگزینی محصول
  - از دست دادن اعتبار

هزینه کیفیت
the cost of quality





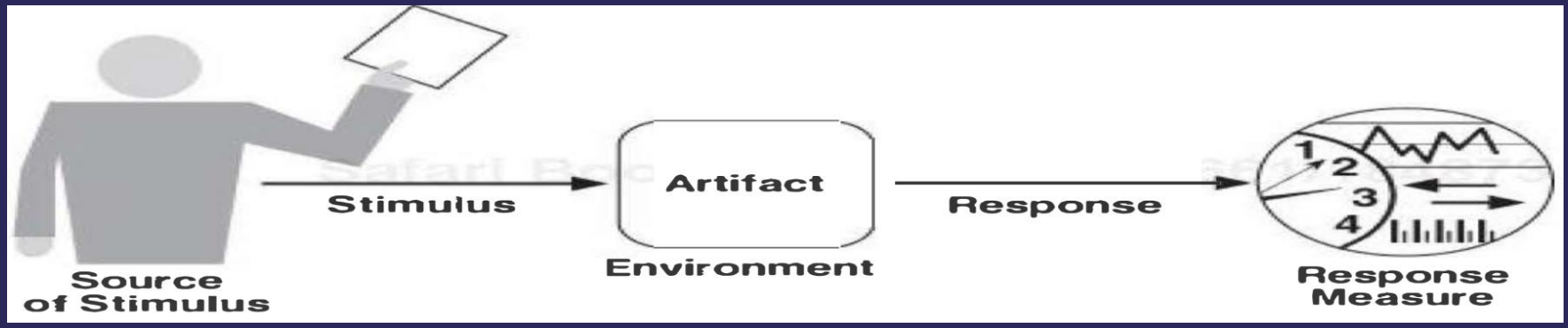
دستیابی به کیفیت

- به کارگیری روش های مهندسی نرم افزار
- تکنیک های مدیریت پروژه
- کنترل کیفیت
- تضمین کیفیت

تضمین کیفیت


تضمین کیفیت
مطابقت با استانداردها
مرور و حسابرسی
تست
تحلیل و جمع آوری error defect
مدیریت تغییرات
مدیریت ریسک
مدیریت امنیت
آموزش

Specifying Quality Attribute Requirements



Specifying Quality Attribute Requirements

- **Stimulus:** We use the term "stimulus" to describe an event arriving at the system. The stimulus can be an event to the performance community, a user operation to the usability community, or an attack to the security community. We use the same term to describe a motivating action for developmental qualities. Thus, a stimulus for modifiability is a request for a modification; a stimulus for testability is the completion of a phase of development.

Specifying Quality Attribute Requirements

- **Stimulus source**: A stimulus must have a source it must come from somewhere. The source of the stimulus may affect how it is treated by the system. A request from a trusted user will not undergo the same scrutiny as a request by an untrusted user.

Specifying Quality Attribute Requirements

- **Response:** How the system should respond to the stimulus must also be specified. The response consists of the responsibilities that the system (for runtime qualities) or the developers (for development-time qualities) should perform in response to the stimulus. For example, in a performance scenario, an event arrives (the stimulus) and the system should process that event and generate a response. In a modifiability scenario, a request for a modification arrives (the stimulus) and the developers should implement the modification without side effects and then test and deploy the modification.

Specifying Quality Attribute Requirements

- **Response measure:** Determining whether a response is satisfactory whether the requirement is satisfied is enabled by providing a response measure. For performance this could be a measure of latency or throughput; for modifiability it could be the labor or wall clock time required to make, test, and deploy the modification.

Specifying Quality Attribute Requirements

- **Environment:** The environment of a requirement is the set of circumstances in which the scenario takes place. The environment acts as a qualifier on the stimulus. For example, a request for a modification that arrives after the code has been frozen for a release may be treated differently than one that arrives before the freeze. A failure that is the fifth successive failure of a component may be treated differently than the first failure of that component.

Specifying Quality Attribute Requirements

- **Artifact:** Finally, the artifact is the portion ofthe system to which the requirement applies. Frequently this is the entire system, but occasionally specific portions of the system may be called out. A failure in a data store may be treated differently than a failure in the metadata store. Modifications to the user interface may have faster response times than modifications to the middleware.

بیان مشخصات نیازمندی های کیفی با روش مبتنی بر سناریو
Scenario based approach

performance

Source : کاربران

Stimulus:
درخواست
تراکنش

Artifact : system

Environment: normal mode

response :
پردازش تراکنش

response measure:
با توان عملیاتی 6 تراکنش در دقیقه

security

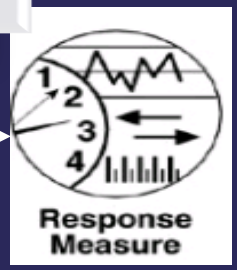
Source : not authorized with access to limited resources

Stimulus: access system services

Artifact : system service

Environment: normal mode

response : restricts availability of services



response measure: With Access controllability 99%