Nickname: mike6606
Age: 8 years 10 months
Fans: 0
Followers: 5
+ Plus follow

< March 2023 >

day	One	Two	Three	Four	Five	Six
26	27	28	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

Search

Look for it

Google search

Frequently used links

My essays
My comment
My involvement
Latest comments
My labels

Essay archives

January 2021 (1)
January 2018 (3)
January 2017 (8)

Read the leaderboard

1. Software Engineering Practitioners' Research Methods Chapter 13 Answers (1578)
2. Software Engineering Practitioners' Research Methods Chapter 1471 Answers (<>)
3. Software Engineering Research Methods for Practitioners Chapter 1290 Answers (<>)
4. Software Engineering Practitioners' Research Methods Chapter 34 Answers (1282)
5. Software Engineering Practitioners' Research Methods Chapter 1264 Answers (<>)

Recommended leaderboards

1. Software Engineering Practitioners' Research Methods Chapter 1 Answers (<>)

Software Engineering Research Methods for Practitioners Chapter 21 Answers

Problem:

Some people say that "variation control is the heart of quality control." Since every program that is created is different from every other program, what are the variations that we look for and how do we control them?

Answer:

4633-16-1P SA: 9420

SR: 6376

A program is equal to data structure plus algorithm. Generally, different programmers will design a program in a different way. Logic of one person varies from the other as per their own thinking. Coding of one programmer will be different from others in solving the same problem.

So, we can expect variations in the design and coding of the data structure and the algorithm of a program by different individuals. Also the programming language chosen by each may vary. Hence we can look for the variations also in the syntax, logic, complexity, readability.

We can control the variations by checking the types and levels of complexity that can be used in the design of the data structures and the algorithms of a program.

Problem:

Is it possible to assess the quality of software if the customer keeps changing what it is supposed to do?

Answer:

Yes, If we define quality as "conformance to requirements," and requirements are dynamic (keep changing), it is possible to assess the quality of software. The definition of quality will be dynamic and an assessment of quality will be difficult

Problem:

Quality and reliability are related concepts but are fundamentally different in a number of ways. Discuss the differences.

Answer:

4633-16-3P SA: 9420

SA: 6376

Quality and reliability both are related concepts. But fundamentally these both are different in some scenarios.

1. Quality control is concerned with the performance of a product at one point in time.

2. Quality focuses on the software's conformance to explicit and implicit requirements.

3. Software quality assurance is the mapping of the managerial principles and design disciplines of quality assurance onto the applicable managerial and technological space of software engineering.

Where as

1. Reliability is concerned with the performance of a product over its entire life time.

2. Reliability focuses on the ability of software to function correctly as a function of time or some other quantity.

3. Software reliability models extend measurements, enabling collected defect data to be extrapolated into projected failure rates and reliability predictions.

Problem:

Can a program be correct and still not be reliable? Explain.

Answer:

Reliability:-

Yes, it is possible for a program to conform all explicit functional and performance requirements at a given instant. Because, reliability uses statistical analysis to determine the software failure with likelihood occur.

Can a program be correct and still not exhibit good quality? Explain.

Answer:

Yes, it is possible for a program to conform to all explicit functional and performance requirements at a given instant, yet to not perform well during maintenance, integration on other systems, or might have hardware dependencies.

Many programs have been patched together, so that they work yet these programs exhibit very low quality if measured by McCall's quality factors.

Problem:

Why is there often tension between a software engineering group and an independent software quality assurance group? Is this healthy?

Answer:

Software Quality Assurance group vs. Software Engineering group:

Between these two groups, there is a common natural tension because, if the SQA group takes on the role of the "observing the bugs of software code," flagging quality problems and high-lighting shortcomings in the developed software, this is normal and this would not be embraced with open arms by the software engineering group.

This software quality assurance group helps ensure that they fit the project’ s needs and verifies that they will be usable for performing the necessary reviews and audits throughout the software life cycle.

As long as the tension does not degenerate into hostility, there is no problem. It is important to note, however, that a software engineering organization should work to eliminate this tension by encouraging a team approach that has development and QA people working together toward common goal of high quality software.

Problem:

You have been given the responsibility for improving the quality of software across your organization. What is the firs thing that you should do? What’ s next?

Answer:

For improving the quality of software across the organization, First of all i evaluate the quality Based on the formal technical reviews, I checkout the total working process of that software. If it is working smoothly, then the number of SQA activities might be implemented. That is

Change control and SCM;

Comprehensive testing methodology;

SQA audits of documentation and related software.

In this process software metrics can help.

Problem:

Besides counting errors and defects, are there other countable characteristics of software that imply quality? What ar they and can they be measured directly?

Answer:

Suppose maintainability measured by mean – time – to – change; portability as measured by an index that indicates conformance to language standard, portability complexity, availability , reliability.

Portability as measured by an index that indicates conformance to language standard; complexity as measured by McCabe's metric, and so on

Here reliability focuses on the ability of software to function correctly as a function of time or some other quantity.

Safety considers the risks associated with failure of a computer based system that is controlled by software.

Quality focuses on the software’ s conformance to explicit and implicit requirements.

In most of the cases an assessment of quality considers may factors that are qualitative in nature.

Problem:

The MTBF concept for software is open to criticism. Explain why.

Answer:

4633-16-9P SA: 9420

SR: 6376

MTBF (mean-time-between-failure) is a simple measure of reliability.

MTTF stands for Mean-Time-To-Failure and

MTTR for Mean-Time-To-Repair

MTBF is related to the mean times. The concept of MTBF is open to criticism because of various reasons. Its measures are based on CPU time, not wall clock time. One of the reasons is it can be problematic for two reasons. They are

1. It projects a time span between failures, but does not provide us with a projected failure rate, and

2. MTBF can be misinterpreted to mean average life span even though this is not what it implies.

Problem:

Consider two safety-critical systems that are controlled by computer. List at least three hazards for each that can be directly linked to software failures.

Answer:

To build safety – critical systems, instead of simply trying to get software correct and assuming that will ensure. System safety, attention is focused on eliminating or controlling the software behaviors.

1. The software requirements are complete and specify only safe behaviors.

2. The entire software development and maintenance process eliminates (or) reduces the possibility of the unsafe behavior.

Software hazard analysis is a form of subsystem hazard analysis used to identify safety – critical software behavior.

The system safety design constrains and information from the system hazard analysis, is used to:

1. Develop software safety for design constraints.

2. Identify specific software safety requiems.

3. Device software and system safety test plans and testing requirements.

4. Trace safety related requirements

5. Develop safety – related information for operations, maintenance.

Problem:

Acquire a copy of ISO 9001:2000 and ISO 9000-3. Prepare a presentation that discusses three ISO 9001 requirements and how they apply in a software context.

Answer:

ISO 9001: 2000 is the generic source of requirements for quality assurance in design, development, production; installation and servicing as well as the standard against which quality management systems for software are assessed.

In ISO 9000:2000 quality management system – fundamentals and vocabulary.

ISO 9004: 2000 quality management systems

- Guidelines for performance imprisonment

ISO 9000: 2000 explains the principles underlying the change' s ISO 9001 and defines vocabulary.

ISO 9001:2000 is the quality assurance standard the applies software engineering. The standard contains 20 requirements that must be present for an effective quality assurance system.

Because the ISO 9001: 2000 standard is applicable to all engineering disciplines, a special set of ISO 9000 -3 have been developed to help interpret standard for use in the software process.

Solution: Chapter 21: SOFTWARE QUALITY ASSURANCE

21.1 We look for variation in the traceability from requirements to design and design to code. We assess the variation in the form and content of work products. We look for variation in the process — repeatable process is a goal. We look for variation in expected and actual results derived from software testing.

21.3 Quality focuses on the software's conformance to explicit and implicit requirements. Reliability focuses on the ability of software to function correctly as a function of time or some other quantity. Safety considers the risks associated with failure of a computer-based system that is controlled by software. In most cases an assessment of quality considers many factors that are qualitative in nature. Assessment of reliability and to some extent safety is more quantitative, relying on statistical models of past events that are coupled with software characteristics in an attempt to predict future operation of a program.

21.4 Yes. It is possible for a program to conform to all explicit functional and performance requirements at a given instant, yet have errors that cause degradation that ultimately causes the program to fail.

21.5 Absolutely. Many programs have been patched together or otherwise "kludged up" so that they work yet these programs exhibit very low quality if measured by McCall' s quality factors.

21.6 There is often a natural "tension" that exists between these two groups. The reason is simple: if the SQA group takes on the role of the "watch dog," flagging quality problems and high-lighting shortcomings in the developed software, it is only normal that this would not be embraced with open arms by the software engineering group. As long as the tension does not degenerate into hostility, there is no problem. It is important to note, however, that a software engineering organization should work to eliminate this tension by encouraging a team approach that has development and QA people working together toward a common goal—high quality software.

21.7 Institute formal technical reviews. After these are working smoothly, any of a number of SQA activities might be implemented: change control and SCM; comprehensive testing methodology; SQA audits of documentation and related software. Also software metrics can help.

21.8 Any countable measure that indicates the factors noted that are candidates. For example, maintainability as measured by mean-time-to-change; portability as measured by an index that indicates conformance to language standard; complexity as measured by McCabe's metric, and so on.

> 21.9 For hardware the MTBF concept is based on statistical error data that occurs due to physical wear in a product. In general, when a failure does occur in hardware, the failed part is replaced with a spare. However, when an error occurs for software, a design change is made to correct it. The change may create side effects that generate other errors. Therefore, the statistical validity of MTBF for software is suspect.
>
> 21.10 Classic examples include aircraft avionics systems, control systems for nuclear power plants, software contained in sophisticated medical instrumentation (e.g., CAT scanners or MRI devices) control systems for train or subway systems; elevator control systems.

21.11 See the SEPA Web site for links to ISO sites.

mike6606
followers - 0 followers - 5

0 0

+ Plus follow

« Previous: Software Engineering Research Methods for Practitioners Chapter 20 Answers
» Next: Software Engineering Research Methods for Practitioners Chapter 26 Answers

posted @ 2021-01-22 00:50 mike6606 read (542) Comments (0) Edit Favorite repor

Editor's Choice:
· Modern Image Performance Optimization - Fault Tolerance and Accessibility of Image Resources
· Talk: The Evolution of Service Architecture
· I'm on timeout with Redis again
· Clever use of CSS variables to create high-end grid animations
· Record the optimization of a lock

Reading Ranking:
· 3 mistakes to avoid when using Vue 10
· ASP.NET Core Web API interface throttling
· 【Fault Bulletin】CC attack is coming again, making March worse ·
Modern Image Performance Optimization and Experience Optimization Guide - Fault tolerance and accessibility of image resources
· How to JWT authenticate and authorize WebAPI in Net6.0