Erlyn S. Andrin

BSIT – NT 4101

Systems Quality and Assurance

1. List and briefly describe the various causes of software errors.

   Answer:

   - **Faulty definition of requirements**

   The faulty definition of requirements, usually prepared by the client, is one of the main causes of software errors. The most common errors of this type are erroneous definition of requirements, absence of vital requirements, incomplete definition of requirements and inclusion of unnecessary requirements.

   - **Client-developer communication failures**

   Misunderstandings resulting from defective client-developer communication are additional causes of errors. Typical situations: Misunderstanding of the client's instructions relating to the requirement document and to changes requested either written or orally by the client. Additional misunderstandings are failures to understand and to give the needed attention to the client's response to design problems raised by the development team.

   - **Deliberate deviations from software requirement**

   In several circumstances, developers may deliberately deviate from the documented requirements, often causing software errors. Common situations of this type: reuse of software modules taken from an earlier project, omission of part of the required functions in an attempt to cope with time or budgetary pressures and developer-initiated improvements, introduced without the client's approval.

   - **Logical design errors**

   Software errors of this type are mainly failures of systems architects, software engineers, systems analysts, etc., to formulate the software requirements into the proper algorithms, boundary conditions, omission of required system states, etc.

   - **Coding errors**

   The reasons that cause programmers to make coding errors include misunderstanding the design documentation, linguistic errors, errors in the application of CASE, other development tools, and so forth.

   - **Noncompliance with documentation and coding instructions**

   One may ask why noncompliance with coding instructions should cause software errors. It is believed that "non-complying" software is expected to increase the rate of errors made by development and maintenance teams. It is due to erroneous understanding by other members of the development team and maintenance team when performing corrections or changes.

- **Shortcomings of the testing process**

Shortcomings of the testing process affect the error rate by leaving a greater number of undetected or uncorrected errors. These shortcomings result from: incomplete test plans, failure to document and report detected errors and faults, failure to promptly correct detected software faults and incomplete correction of detected errors.

- **Procedure errors**

Procedure errors, especially in complex software systems, may incorrectly direct the user with respect to the activities required.

- **Documentation errors**

The documentation errors that trouble the development and maintenance teams are errors in the design documents and in the documentation integrated into the body of the software. These errors can cause additional errors in further stages of development and during maintenance.

2. Classify the causes of error according to the groups responsible for the error:

*the client's staff,

Answer: Faulty definition of requirements and Client-developer communication failures.

*the systems analysts,

Answer: Client-developer communication failures, Deliberate deviations from software requirements, Logical design errors, Noncompliance to documentation and coding instructions, Procedure errors and Documentation errors.

*the programmers/ Developers

Answer: Coding errors, Noncompliance to documentation and coding instructions and Documentation errors.

*the testing staff/ Quality Assurance staff

Answer: Shortcomings of the testing process and Procedure errors.

*or is it a shared responsibility belonging to more than one group?

Answer: Yes

3. What are the differences between the IEEE definition of SQA and the expanded definition used in our eBook (Daniel Galin)?

Answer:

- **Life cycle phase**.

The IEEE SQA definitions referring to the software are limited to the development phase, while the expanded SQA definition covers the whole software life cycle, namely the development phase and maintenance during operation phase.

- **Subjects included.**

The IEEE SQA definitions are limited to the technical aspects of the functional requirements, while the expanded SQA definition also includes activities dealing with scheduling and the budget of software development and maintenance.

4. According to the IEEE definition of SQA, quality control (QC) is not equated with quality assurance (QA).

In what respects does QC vary from QA?

Answer:

Both QC and QA have a common objective, namely, to bring the quality of a software product to a required controlled level. However, QA has an additional important objective to achieve the aforementioned basic objective, while minimizing the costs of quality. Thus, QC varies from QA by the scope of tools applied. QC focuses on evaluating the quality of a developed product by inspection and testing, and withholding of the software product until the detected errors are repaired and the software product qualifies.

Why can QC be considered part of QA?

Answer:

Evaluation activities of the software product that comprise QC are a major part of QA the other major part being prevention activities, consequently QC is a part of QA.