CRJ2A3 Penjaminan Mutu Perangkat Lunak

Introduction Software Quality Assurance

Tim Dosen SQA KK SE

Program Studi S1 Rekayasa Perangkat Lunak
Universitas Telkom
2022

1. Define software, software quality and software quality assurance.

What is software?

IEEE definition of Software is:

Computer programs, procedures, documentation and data pertaining to the operation of a computer system.

Quality - definition

- The American Heritage Dictionary defines quality as a **characteristic** or **attribute of something**
- Refers to measurable characteristics that we can compare to known **standards**
- Quality means **conformance to requirements** *(Crosby, 1979)*

Software Quality –definition

Software quality is: [IEEE Definition]

1. The degree to which a system, component, or process **meets specified requirements**.

2. The degree to which a system, component, or process **meets user needs or expectations**.

Software Quality Assurance

Software quality assurance – The IEEE definition

1. **A planned and systematic pattern** of all actions necessary to provide adequate confidence that a product conforms to established technical requirements.
2. **A set of activities designed** to evaluate the process by which the products are developed or manufactured.

SQA – expanded definition

Software quality assurance is:

A systematic, planned set of actions necessary to provide adequate confidence that the software development process or the maintenance process of a software system product conforms to established functional technical requirements as well as with the managerial requirements of keeping the schedule and operating within the budgetary confines.

(ISO 9000 standards regarding SQA, the Capacity Maturity Model (CMM) for software (Paulk *et al.*, 1993; Tingey, 1997).

The main deviations from the IEEE definition

- SQA should not be limited to the development process.
- SQA actions should not be limited to the technical aspects of the functional requirements, but should include also activities that deal with scheduling and the budget.

The Relationship between Software Quality Assurance and Software Engineering

- **Software engineering** is the application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software.
- The characteristics of software engineering, especially its systematic, disciplined and quantitative approach, make software engineering a good environment for achieving SQA objectives.
- It is commonly accepted that cooperation between software engineers and the SQA team is the way to achieve efficient and economic development and maintenance activities that, at the same time, assure the quality of the products of these activities.

2. Objectives of SQA activities

Objective of SQA Activities

1. Software development phase (process-oriented):
2. Software maintenance phase (product-oriented):

Objective SQA in SW Development :

1. Assuring an acceptable level of confidence that the software will conform to functional technical requirements.
2. Assuring an acceptable level of confidence that the software will conform to managerial scheduling and budgetary requirements.
3. Initiating and managing of activities for the improvement and greater efficiency of software development and SQA activities.

Objective SQA in SW Maintenance :

1. Assuring with an acceptable level of confidence that the software maintenance activities will conform to the functional technical requirements.
2. Assuring with an acceptable level of confidence that the software maintenance activities will conform to managerial scheduling and budgetary requirements.
3. Initiating and managing activities to improve and increase the efficiency of software maintenance and SQA activities.

3. Difference of Error, Faults and Failures

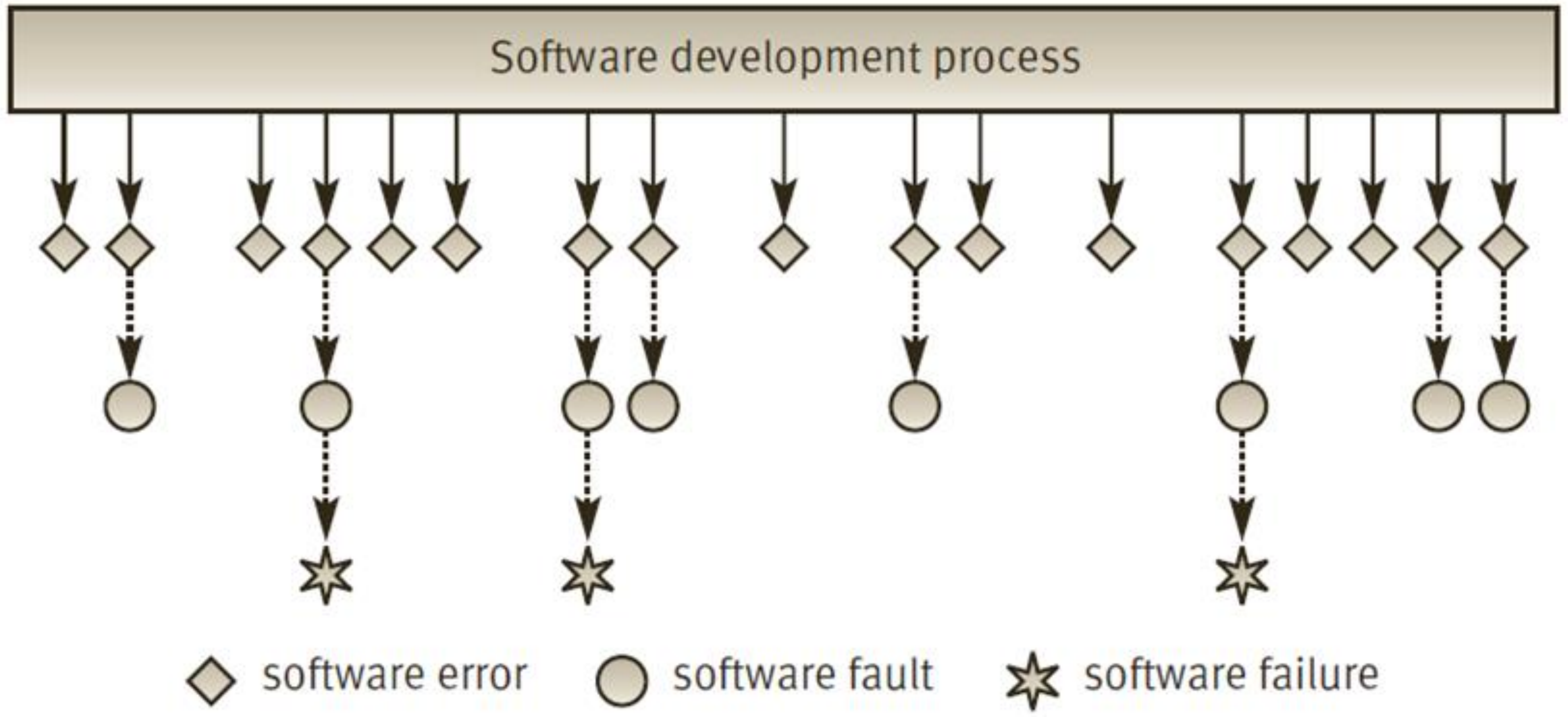
Software errors, faults and failures

A. Software errors are sections of the code that are partially or totally incorrect as a result of a grammatical, logical or other mistake. (made by a systems analyst, a programmer, or another member of the software development team) **made by human**

B. Software faults are software errors that **cause the incorrect functioning** of the software during a specific application.

C. Software faults become software failures only when they are "activated", that is, when a user tries to apply the specific software section that is faulty. Thus, the root of any software failure is a software error.

The Relationships between Software errors, faults and failures



4. Identify various causes of SW Error

Identify the various causes of software errors

1. **Coding errors**
2. **Logical design errors**
3. **Faulty requirements definition**
4. **Client–developer communication failures**
5. **Deliberate deviations from software requirements**
6. **Non-compliance with documentation and coding instructions**
7. **Shortcomings of the testing process**
8. **Procedure errors**
9. **Documentation errors**

It should be emphasized that all causes of error are human, the work of systems analysts, programmers, software testers, documentation experts, and even clients and their representatives.

Coding Errors

- A broad range of reasons cause programmers to make coding errors.
- These include misunderstanding the design documentation, linguistic errors in the programming languages, errors in the application of CASE and other development tools, errors in data selection, and so forth.

Logical design errors

- Definitions that represent software requirements by means of erroneous algorithms.
- Process definitions that contain sequencing errors.
- Erroneous definition of boundary conditions.
- Omission of required software system states.
- Omission of definitions concerning reactions to illegal operation of the software system.

Faulty requirements definition

- Erroneous definition of requirements.
- Absence of vital requirements.
- Incomplete definition of requirements.
- Inclusion of unnecessary requirements, functions that are not expected to be needed in the near future.

Client–developer communication failures

- Misunderstanding of the client’s instructions as stated in the requirement document.
- Misunderstanding of the client’s requirements changes presented to the developer in written form during the development period.
- Misunderstanding of the client’s requirements changes presented orally to the developer during the development period.
- Misunderstanding of the client’s responses to the design problems presented by the developer.
- Lack of attention to client messages referring to requirements changes and to client responses to questions raised by the developer on the part of the developer.

Deliberate deviations from software requirements

- The developer reuses software modules taken from an earlier project without sufficient analysis of the changes and adaptations needed to correctly fulfill all the new requirements.
- Due to time or budget pressures, the developer decides to omit part of the required functions in an attempt to cope with these pressures.
- Developer-initiated, unapproved improvements to the software, introduced without the client's approval, frequently disregard requirements that seem minor to the developer. Such "minor" changes may, eventually, cause software errors.

Non-compliance with documentation and coding instructions

- Team members who need to coordinate their own codes with code modules developed by “non-complying” team members can be expected to encounter more than the usual number of difficulties when trying to understand the software developed by the other team members.
- Individuals replacing the “non-complying” team member (who has retired or been promoted) will find it difficult to fully understand his or her work.
- The design review team will find it more difficult to review a design prepared by a non-complying team.
- The test team will find it more difficult to test the module; consequently, their efficiency is expected to be lower, leaving more errors undetected.
- Maintenance teams required to contend with the “bugs” detected by users and to change or add to the existing software will face difficulties when trying to understand the software and its documentation.

Shortcomings of the testing process

- Incomplete test plans leave untreated portions of the software or the application functions and states of the system.
- Failures to document and report detected errors and faults.
- Failure to promptly correct detected software faults as a result of inappropriate indications of the reasons for the fault.
- Incomplete correction of detected errors due to negligence or time pressures.

Procedure errors

- Procedures direct the user with respect to the activities required at each step of the process.
- They are of special importance in complex software systems where the processing is conducted in several steps, each of which may feed a variety of types of data and allow for examination of the intermediate results.

Documentation Errors

- Omission of software functions.
- Errors in the explanations and instructions given to users, resulting in “dead ends” or incorrect applications.
- Listing of non-existing software functions, that is, functions planned in the early stages of development but later dropped, and functions that were active in previous versions of the software but cancelled in the current version.

TERIMA KASIH dan SELAMAT BELAJAR