

QUALITY ASSURANCE IN SOFTWARE PRODUCTION

A decorative graphic at the bottom of the slide consists of a solid green trapezoidal shape pointing downwards, which is overlaid on a yellow trapezoidal shape pointing upwards. The two shapes meet at a white, jagged, V-shaped line that creates a central point at the bottom.

THE UNIQUENESS OF SOFTWARE QUALITY ASSURANCE

- ▶ High product complexity:
there are millions of software operation possibilities
- ▶ Product visibility:
Software products are "invisible"
- ▶ Opportunities to detect defects ("bugs") are limited to the product development phase

THE UNIQUENESS OF SQA (CONT.)

Table 1.1: Factors affecting defect detection in software products vs. other industrial products

Characteristic	Software products	Other industrial products
Complexity	Usually, very complex product allowing for very large number of operational options	Degree of complexity much lower, allowing at most a few thousand operational options
Visibility of product	Invisible product, impossible to detect defects or omissions by sight (e.g. of a diskette or CD storing the software)	Visible product, allowing effective detection of defects by sight
Nature of development and production process	Opportunities to detect defects arise in only one phase, namely product development	Opportunities to detect defects arise in all phases of development and production: <ul style="list-style-type: none">■ Product development■ Product production planning■ Manufacturing

TYPICAL CAUSES OF SOFTWARE ERRORS

- ▶ Faulty definition of requirements
- ▶ Client-developer communication failures
- ▶ Deliberate deviations from software requirements
- ▶ Logical design errors
- ▶ Coding errors
- ▶ Non-compliance with documentation and coding instructions
- ▶ Shortcomings of the testing process


FAULTY DEFINITION OF REQUIREMENTS

- ▶ One of the main causes of software errors
- ▶ Erroneous definition of requirements
- ▶ Missing vital requirements
- ▶ Incomplete requirements
- ▶ Unnecessary requirements

CLIENT-DEVELOPER COMMUNICATION FAILURES

- ▶ Misunderstanding of the client's requirement document
- ▶ Miscommunications during client-developer meetings
- ▶ Misinterpretation of the client's requirement changes


DELIBERATE DEVIATIONS FROM THE SOFTWARE REQUIREMENTS

- ▶ The developer reuses software modules from earlier projects without sufficient analysis of the changes needed to fulfill the requirements
 - ▶ Due to time/money pressures, the developer leaves parts of the required functions in an attempt to manage the pressure
 - ▶ The developer makes unapproved improvements to the software without the client's knowledge
- 

LOGICAL DESIGN ERRORS

- ▶ Defining the software requirements by means of erroneous algorithms
- ▶ Process definitions contain sequencing errors
- ▶ Erroneous definition of boundary conditions
- ▶ Omission of definitions concerning special cases (lack of error handling and special state handling)

CODING ERRORS


- ▶ Misunderstanding of the design documents,
 - ▶ linguistic errors,
 - ▶ development tool errors,
 - ▶ data errors,
 - ▶ data representation error,
 - ▶ etc.
- 

NON-COMPLIANCE WITH DOCUMENTATION AND CODING INSTRUCTIONS

Non-compliance with coding/documentation standards makes it harder to understand, review, test and maintain the software

- ▶ Unusual number of problems to cope with
- ▶ Retired or promoted team member
- ▶ Lack of understanding the design

SHORTCOMINGS OF THE TESTING PROCESS

- ▶ Greater number of errors are left undetected and uncorrected
 - ▶ Incomplete test plans will leave many of the functions and states untested
 - ▶ Failures to report and promptly correct found errors and faults
 - ▶ Incomplete test due to time pressures
- 

SOFTWARE DEFINITION

- ▶ IEEE definition.

Software is:

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

- ▶ ISO definition

(ISO, 1997, Sec. 3.11 and ISO/IEC 9000-3 Sec. 3.14), lists the following four components of software:

- ▶ Computer programs (the “code”)
- ▶ Procedures
- ▶ Documentation
- ▶ Data necessary for operating the software system.

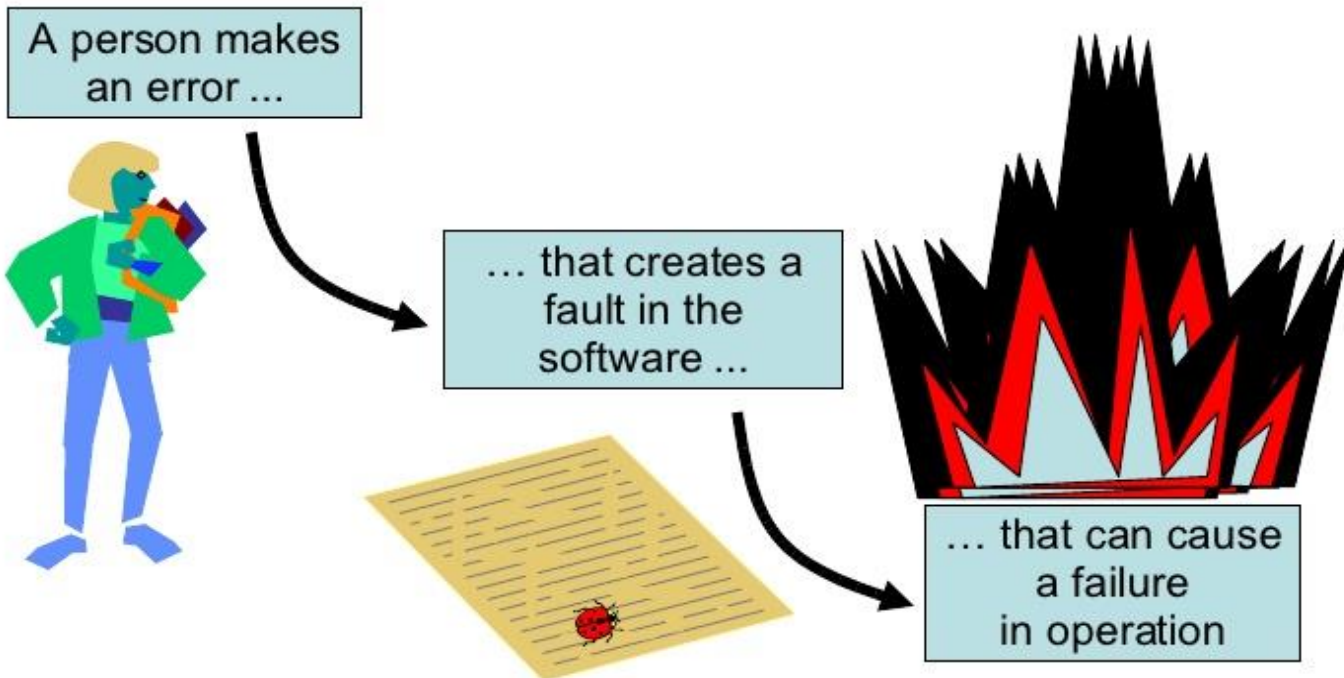
SOFTWARE ERRORS, FAULTS AND FAILURES

Software error

Software fault

Software failure

Error - Fault - Failure



SOFTWARE QUALITY – DEFINITION

Software quality is:

1. The degree to which a system, component, or process meets specified requirements.
2. The degree to which a system, component, or process meets customer or user needs or expectations.

Software quality is defined as:


Conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software.

SOFTWARE QUALITY ASSURANCE DEFINITION

„SQA 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”

--Daniel Galin

THE OBJECTIVES OF SQA ACTIVITIES

- ▶ Software development (process oriented)
 - ▶ To assure that the software will meet the functional technical requirements
 - ▶ To assure that the software will conform to managerial scheduling and budgetary requirements
 - ▶ To continuously improve the development process and SQA activities in order to improve the quality and at the same time reduce the cost
 - ▶ The same things apply to software maintenance (product oriented)
- 

THE OBJECTIVES OF SQA ACTIVITIES

Software maintenance (product oriented)

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. This involves improving the prospects of achieving functional and managerial requirements while reducing costs.

SOFTWARE QUALITY FACTORS

McCall's quality factor model consisting of 11 quality factors grouped into 3 categories

- ▶ Product operation factors:

- ▶ Correctness,
- ▶ Efficiency,
- ▶ Integrity,
- ▶ Usability

- ▶ Product revision factors:

- ▶ Maintainability,
- ▶ Flexibility,
- ▶ Testability

- ▶ Product transition factors:

- ▶ Portability,
- ▶ Reusability,
- ▶ Interoperability

PRODUCT OPERATION FACTORS

Correctness: defined in a list of the software system's required output

- ▶ The output mission (e.g. red alarms when temperature rises to 100 °C)
- ▶ Required accuracy of the output (e.g. non-accurate output will not exceed 1%)
- ▶ Completeness of the output info (e.g. probability of missing data less than 1%)
- ▶ The up-to-dateness of the info (e.g. it will take no more than 2s for the information to be updated)
- ▶ The availability of the info (e.g. reaction time for queries will be less than 2s on average)
- ▶ The required standards and guidelines (the software and its docs must comply with the client's guidelines)

PRODUCT OPERATION FACTORS (CONT.)

- ▶ Efficiency:

- ▶ Focus is on hardware resources needed to perform the operations to fulfill the requirements

- ▶ memory,

- ▶ storage,

- ▶ CPU speed,

- ▶ data communication capability,

- ▶ battery, etc.)

PRODUCT OPERATION FACTORS (CONT.)

- ▶ Reliability:
 - ▶ Determines the maximum allowed software system failure rate
 - ▶ Can focus on entire system or just on a separate function
 - ▶ E.g. a heart-monitor's failure rate must be less than 1:20 years

PRODUCT OPERATION FACTORS (CONT.)

- ▶ Integrity:
 - ▶ Security:
 - ▶ Requirements to prevent access to unauthorized persons („read permit”)
 - ▶ Rights management (e.g. limit the ”write permit” to key personnel)
 - ▶ Safety:
 - ▶ recorded exactly as intended (rejecting mutually exclusive access)
 - ▶ Data integrity
 - ▶ Entity integrity (primary keys)
 - ▶ Referential integrity (foreign keys)
 - ▶ Domain integrity (set of values to use)
 - ▶ User defined integrity

PRODUCT OPERATION FACTORS (CONT.)

- ▶ Usability:

- ▶ Deals with the scope of staff resources needed to train a new employee and to operate the software system
- ▶ E.g. training of a new employee to operate the system will take no more than 2 working days

PRODUCT REVISION FACTORS

- ▶ Maintainability:
 - ▶ Determines the effort needed to identify the causes for software failures, to correct them, and to verify the success of the corrections
 - ▶ Refers
 - ▶ to the modular structure of the software as well as
 - ▶ to the manuals and
 - ▶ documentations
- ▶ E.g.:
 - ▶ Size of the module should not exceed a page on the screen
 - ▶ Coding standards to apply

PRODUCT REVISION FACTORS (CONT.)

- ▶ Flexibility:

- ▶ The effort required to support adaptive maintenance activities

- ▶ E.g. man-days required to adapt a software package to a variety of customers of the same trade

- ▶ Testability:

- ▶ Testability requirements include automatic diagnostics checks and log files, etc.

- ▶ E.g. a standard test must be run every morning before the production begins

PRODUCT TRANSITION FACTORS

- ▶ Portability:
 - ▶ Adaptation of the system to other environments of different hardware, OS, etc.
- ▶ Reusability:
 - ▶ Mostly the developer will initiate the reusability requirement by recognizing the potential benefit of a reuse
 - ▶ It's expected to save development resources, shorten the development time and provide higher quality modules

PRODUCT TRANSITION FACTORS (CONT.)

- ▶ Interoperability:
 - ▶ Focuses on developing interfaces with other software systems or with other equipment firmware
 - ▶ E.g.: a laboratory equipment is required to process its results (output) according to a standard data structure, which the laboratory information system can then use as an input

THE SQA SYSTEM

- ▶ Goal is to minimize the number of software errors and to achieve an acceptable level of software quality
- ▶ Can be divided into to six classes:
 - ▶ Pre-project components
 - ▶ Components of project life cycle activities assessment
 - ▶ Components of infrastructure error prevention and improvement
 - ▶ Components of software quality management
 - ▶ Components of standardization, certification, and SQA system assessment
 - ▶ Organizing for SQA-the human components

PRE-PROJECT COMPONENTS

- ▶ The schedule and budget as well as other project commitments are adequately planned
- ▶ Must assure that development and quality plans are correctly determined


COMPONENTS OF PROJECT LIFE CYCLE ACTIVITIES ASSESSMENT

- ▶ Two phases:
 - ▶ Development life cycle (verification-validation-qualification, reviews, expert opinions, software testing)
 - ▶ Operation-maintenance stage (Special maintenance components and life cycle components for improving maintenance tasks)
- ▶ Assuring the quality of parts made by subcontractors and other external participants during development and maintenance phases

COMPONENTS OF INFRASTRUCTURE ERROR PREVENTION AND IMPROVEMENT

- ▶ Goals are to reduce software fault rates and to improve productivity
- ▶ Procedures and work instructions, staff training, configuration management, documentation control, etc.
- ▶ Applied throughout the entire organization


COMPONENTS OF SOFTWARE QUALITY MANAGEMENT

- ▶ Major goals are to control development and maintenance activities and early managerial support (minimize schedule and budget failures)
 - ▶ Software quality metrics, quality cost, project progress control, etc.
- 

COMPONENTS OF STANDARDIZATION, CERTIFICATION, AND SQA SYSTEM ASSESSMENT

- ▶ Implements international, professional and managerial standards within the organization
- ▶ Quality management standards: focuses on *what* is required in regards of managerial quality system (e.g. ISO 9001, SEI CMM assessment standard)
- ▶ Project process standards: methodological guidelines (“how”) for the development team (e.g. IEEE 1012, ISO/IEC 12207)

ORGANIZING FOR SQA – THE HUMAN COMPONENT

- ▶ The organizational base: managers, testing personnel, SQA team, etc.
 - ▶ To develop and support the implementation of SQA components
 - ▶ To detect deviations from SQA procedures and methodology
 - ▶ To suggest improvements to SQA components
- 

REVIEWS IN SQA

- ▶ Sometimes called Formal Technical Review (FTR), Formal Design Review, Inspection, Walkthrough, Peer Review, etc.
- ▶ Originally developed by Michael Fagan in the 1970's (IBM)
- ▶ Meeting technique: based on teamwork
- ▶ The goal is to find errors from basically any written document (specification, code, etc.)


GOALS OF REVIEWS

- ▶ Basic idea is to remove the errors in the early part of the project
- ▶ Project is divided into intermediate stages/phases (makes the progression of the project more visible)
- ▶ Some basic objectives:
 - ▶ Uncover errors in functions, logic or implementation
 - ▶ To verify that the document (software code, specification, etc.) meets its requirements
 - ▶ To ensure that the software has been represented according to the pre-defined standards
 - ▶ To make projects more manageable

ORGANIZING THE REVIEW MEETING

- ▶ The amount of material to be inspected must be reasonable (not too much material)
- ▶ No unfinished material
- ▶ Participants must have enough time to get to know the material beforehand
- ▶ The amount of participants must be kept as low as possible (no unnecessary people)

THE ACTUAL REVIEW MEETING

- ▶ Roles: presenter (usually the producer himself), moderator/review leader, secretary/recorder.
 - ▶ Focus is on finding the problems rather than solving them
 - ▶ Review the product, not the producer.
 - ▶ Limit the amount of debate
 - ▶ The more new errors found the more successful the meeting is
- 

AFTER THE REVIEW MEETING

- ▶ Accept the product (no modifications necessary)
- ▶ Reject the product (severe errors)
- ▶ Accept the product provisionally (small errors, new meeting not necessary)
- ▶ All the findings and conclusions are noted to the record (important)

REVIEWS: PROS/CONS

- ▶ Most of the errors are found early in the project (saves money)
- ▶ Producer has a better understanding of the correctness of his work.
- ▶ All the problems aren't found just by testing: errors in phase products, style errors, unnecessary code, etc.
- ▶ Problems: causes extra work in the early stages of the project, organizing the inspection might be problematic, attitudes, unpreparedness to the meeting

REFERENCES

- ▶ Daniel Galin, *Software Quality Assurance, From theory to implementation. Pearson Education Limited 2004, Essex, England*
- ▶ Roger S. Pressman, Darrel Ince, *Software Engineering: a practitioner's approach, European Adaptation, Fifth Edition. McGraw-Hill Publishing Company, Printed by: Tj International Ltd, Padstow, Cornwall, 2000.*