Introduction to Software Quality Assurance

Based on D. Galin Ch1 – 4, 5 and R. Patton Ch 1

Objectives

- What is Software Quality?
- Why is Software Quality important?
- What is Software Quality Assurance ?
- Software Quality factors
- Elements of Software Quality Assurance
- Development and quality plans
- Process Maturity models

What is Software ?

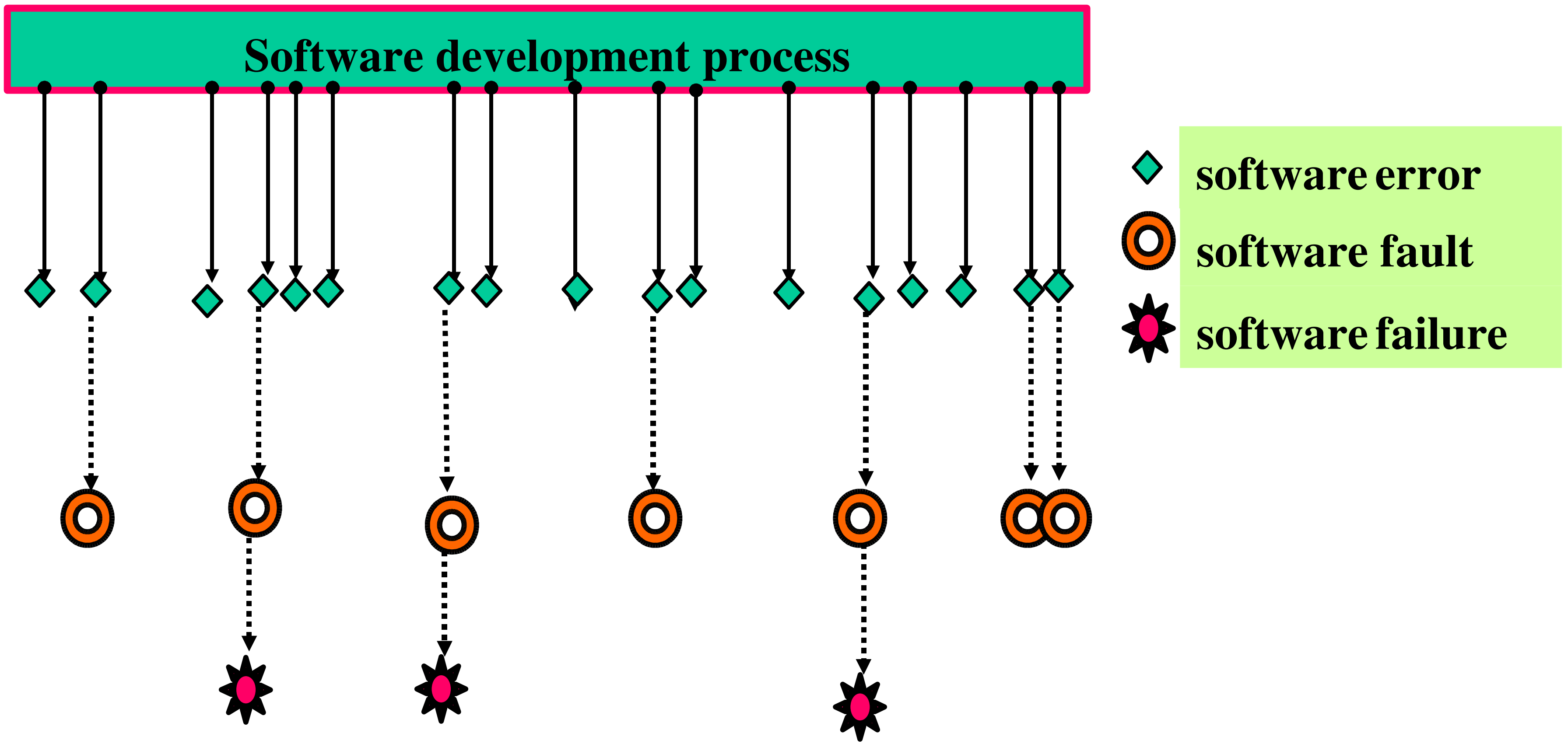
According to the IEEE

Software is:

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

Software Errors, software faults and software failures

- *Bug/defect/fault* consequence of a *human error*
  - results in non-conformance to requirements
  - manifests as *failure* in running software



Nine Causes of Software Errors

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

Development process relation to defects

Majority of defects are introduced in earlier phases

Phase	Percentage of defects	Effort to fix defects
Requirements	56	82
Design	27	13
Code	7	1
Others	10	4

Relative cost of fixing defects

Phase in which found	Cost Ratio
Requirements	1
Design	3 – 6
Coding	10
Unit/Integration testing	15 – 40
System/Acceptance testing	30 – 70
Production	40 – 1000

Requirements are the top reason for a project success or failure

What is Software Quality ?

- *Conformance to requirements*
- Lack of bugs
- Low *defect rate* (# of defects/size unit)
- High *reliability* (number of failures per *n* hours of operation)
    - Measured as Mean Time To Failure (MTTF) probability of failure-free operation in a specified time

What is Software Quality ?

According to the IEEE

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.*

What is Software Quality ?

According to Pressman

Software quality is:

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

Importance of Software Quality

- Software is a major component of computer systems (about 80% of the cost) – used for
  - communication (e.g. phone system, email system)
  - health monitoring,
  - transportation (e.g. automobile, aeronautics),
  - economic exchanges (e.g. e-commerce),
  - entertainment,
  - etc.
- Software defects are extremely costly in term of
  - money
  - reputation
  - loss of life

Importance of Software Quality

- Several historic disasters attributed to software
  - 1988 shooting down of Airbus 320 by the USS Vincennes - cryptic and misleading output displayed by tracking software
  - 1991 patriot missile failure - inaccurate calculation of time due to computer arithmetic errors
  - London Ambulance Service Computer Aided Dispatch System – several deaths
  - On June 3, 1980, the North American Aerospace Defense Command (NORAD) reported that the U.S. was under missile attack.
  - First operational launch attempt of the space shuttle, whose real-time operating software consists of about 500,000 lines of code, failed - synchronization problem among its flight-control computers.
  - 9 hour breakdown of AT&T's long-distance telephone network - caused by an untested code patch

Importance of Software Quality

- Ariane 5 crash June 4, 1996
  - maiden flight of the European Ariane 5 launcher crashed about 40 seconds after takeoff
  - lost was about half a billion dollars
  - explosion was the result of a software error
    - Uncaught exception due to floating-point error: conversion from a 64-bit integer to a 16-bit signed integer applied to a larger than expected number
    - Module was re-used without proper testing from Ariane 4
      - Error was not supposed to happen with Ariane 4
      - No exception handler

Importance of Software Quality

- Mars Climate Orbiter - September 23, 1999
  - Mars Climate Orbiter, disappeared as it began to orbit Mars.
  - Cost about $US 125-million
  - Failure due to error in a transfer of information between a team in Colorado and a team in California
    - One team used English units (e.g., inches, feet and pounds) while the other used metric units for a key spacecraft operation.

Importance of Software Quality

- Mars Polar Lander - December, 1999
  - Mars Polar Lander, disappeared during landing on Mars
  - Failure more likely due to unexpected setting of a single data bit.
    - defect not caught by testing
    - independent teams tested separate aspects

Importance of Software Quality

- Internet viruses and worms
  - Blaster worm ($US 525 millions)
  - Sobig.F ($US 500 millions – 1billions)
- Exploit well known software vulnerabilities
  - Software developers do not devote enough effort to applying lessons learned about the causes of vulnerabilities.
  - Same types of vulnerabilities continue to be seen in newer versions of products that were in earlier versions.
- Usability problems

Importance of Software Quality

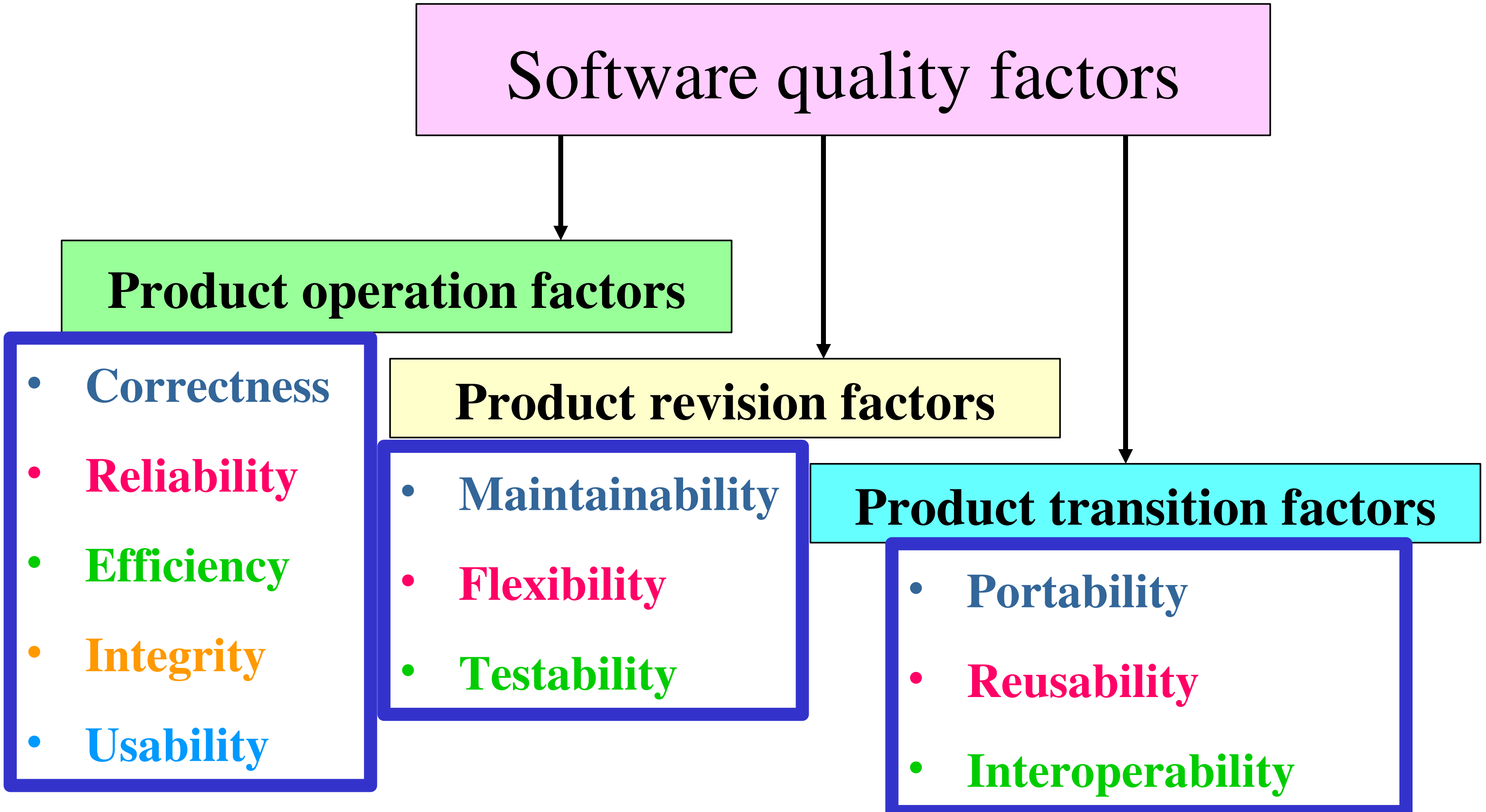
- Monetary impact of poor software quality (Standish group - 1995)
- 175,000 software projects/year - Average Cost per project
  - Large companies - $US 2,322,000
  - Medium companies - $US 1,331,000
  - Small companies - $US 434,000
- 31.1% of projects canceled before completed
  - cost $81 billion
- 52.7% of projects exceed their budget - costing 189% of original estimates
  - cost $59 billion
- 16.2% of software projects completed on-time and on-budget (9% for larger companies)
- Large companies - delivered systems have approximately only 42% of originally-proposed features and functions
- 78.4% of smaller companies projects get deployed with at least 74.2% of their original features and functions.

The Software Quality Challenge

- The uniqueness of the software product
  - High complexity
  - Invisibility of the product
  - Limited opportunities to detect defects (“bugs”)
    - only opportunity is *Product development*
- The environments in which software is developed
  - Contracted
  - Subjection to customer-supplier relationship
  - Requirement for teamwork
  - Need for cooperation and coordination with other development teams
  - Need for interfaces with other software systems
  - Need to continue carrying out a project while the team changes
  - Need to continue maintaining the software system for years

Software Quality Factors

- McCall's software quality factor model



Software Quality Factors

- Correctness
  - accuracy, completeness of required output
  - up-to-dateness, availability of the information
- Reliability
  - maximum failure rate
- Efficiency
  - resources needed to perform software function
- Integrity
  - software system security, access rights
- Usability
  - ability to learn, perform required task

Software Quality Factors

- Maintainability
  - effort to identify and fix software failures (modularity, documentation, etc)
- Flexibility
  - degree of adaptability (to new customers, tasks, etc)
- Testability
  - support for testing (e.g. log files, automatic diagnostics, etc)

Software Quality Factors

- Portability
  - adaptation to other environments (hardware, software)
- Reusability
  - use of software components for other projects
- Interoperability
  - ability to interface with other components/systems

What is Software Quality Assurance?

According to the IEEE

Software quality assurance is:

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

What is Software Quality Assurance?

According to D. Galin

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.

Objectives of SQA in 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) Initiation and management of activities for the improvement and greater efficiency of software development and SQA activities.

Objectives of SQA in maintenance

(1) Assuring an acceptable level of confidence that the software maintenance activities will conform to the functional technical requirements.

(2) Assuring an acceptable level of confidence that the software maintenance activities will conform to managerial scheduling and budgetary requirements.

(3) Initiate and manage activities to improve and increase the efficiency of software maintenance and SQA activities.

Three General Principles of QA

- Know what you are doing
- Know what you should be doing
- Know how to measure the difference

Three General Principles of QA

- Know what you are doing
  - understand what is being built, how it is being built and what it currently does
  - suppose a software development process with
    - management structure (milestones, scheduling)
    - reporting policies
    - tracking

Three General Principles of QA

- Know what you should be doing
  - having explicit requirements and specifications
  - suppose a software development process with
    - requirements analysis,
    - acceptance tests,
    - frequent user feedback

Three General Principles of QA

- Know how to measure the difference
  - having explicit measures comparing what is being done from what should be done
  - four complementary methods:
    - formal methods – verify mathematically specified properties
    - testing – explicit input to exercise software and check for expected output
    - inspections – human examination of requirements, design, code, ... based on checklists
    - metrics – measures a known set of properties related to quality

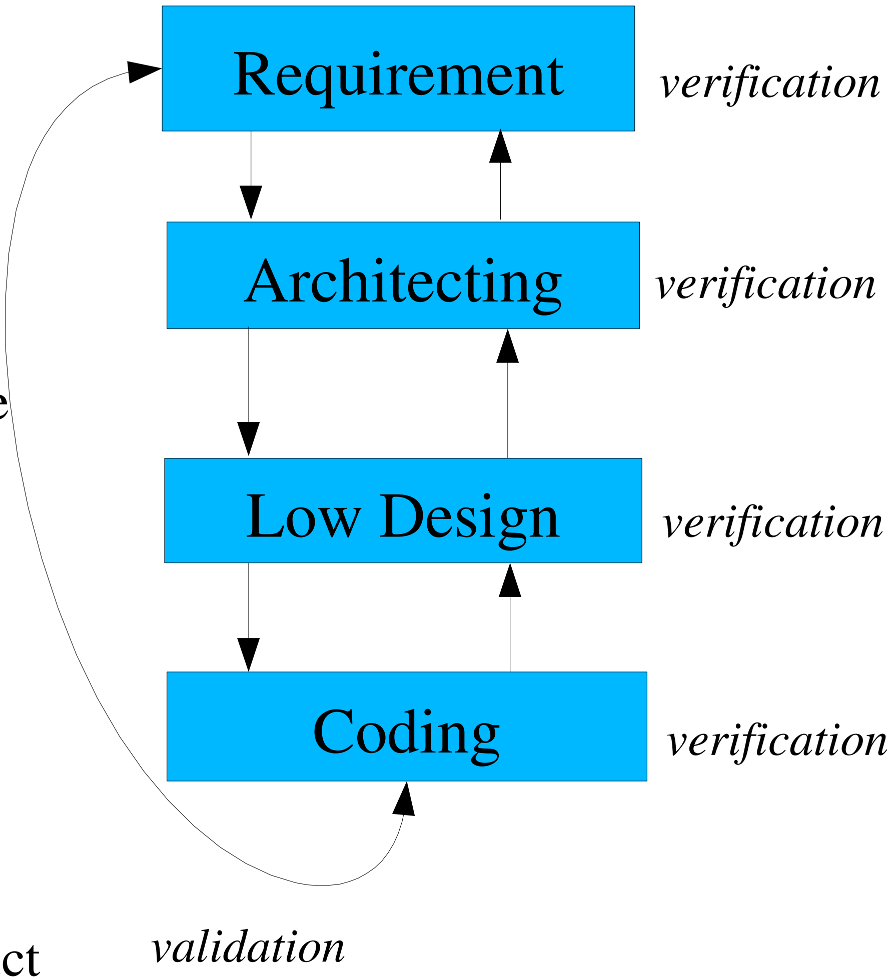
Software Quality Assurance

- SQA: Comprehensive life-cycle approach concerned with every aspect of the software product development process
- Includes
  - comprehensive set of quality objectives
  - measurable quality attributes (quality metrics) to assess progress toward the objectives
  - quantitative certification targets for all component of the software development processes.
- Takes into account:
  - customer product requirements,
  - customer quality requirements, and
  - corporate quality requirements.

SQA

SQE includes

- Verification
  - are we building the product right ?
  - performed at the end of a phase to ensure that requirements established during previous phase have been met
- Validation
  - are we building the right product ?
  - performed at the end of the development process to ensure compliance with product requirements



SQA

SQA includes

- Defect Prevention
  - prevents defects from occurring in the first place
  - Activities: training, planning, and simulation
- Defects detection
  - finds defects in a software artifact
  - Activities: inspections, testing or measuring
- Defects removal
  - isolation, correction, verification of fixes
  - Activities: fault isolation, fault analysis, regression testing

SQA

- Typical activities of a SQA process
  - Requirements validation.
  - Design verification.
  - Static code checking (inspection/reviews).
  - Dynamic testing.
  - Process engineering and standards.
  - Metrics and continuous improvement