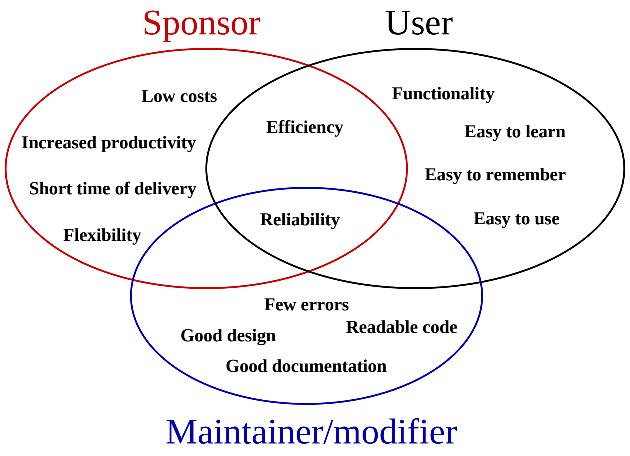
Software Quality Assurance

What is Quality?


Sponsor

User

Maintainer/modifier

Low costs

Increased productivity

Short time of delivery

Flexibility

Efficiency

Reliability

Functionality

Easy to learn

Easy to remember

Easy to use

Few errors

Good design

Readable code

Good documentation

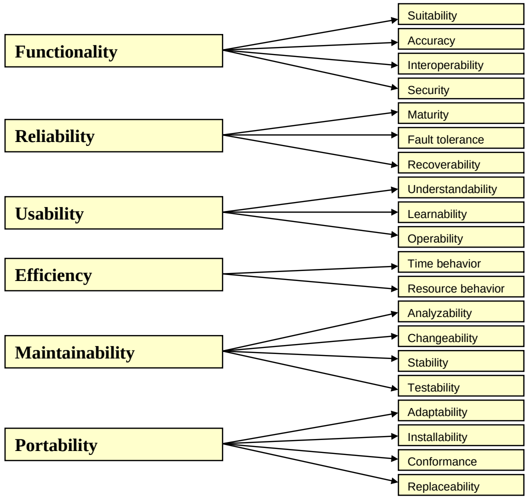
Perspectives on Quality

- Delivered Quality
  - As perceived by the customer
- Reused Quality
  - As perceived by the future developer
- Maintained Quality
  - As perceived by maintenance responsible

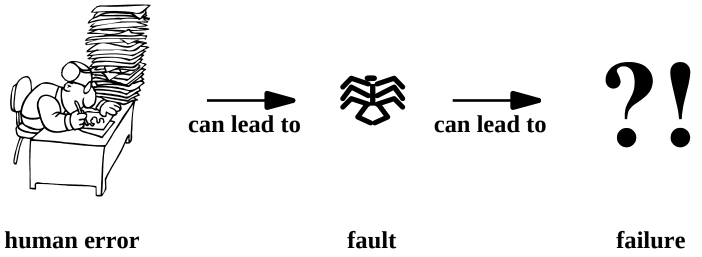
Maximum User Benefit

Minimum Future Development Cost

Quality Factors (ISO 9126)


- **Functionality**
  - Suitability
  - Accuracy
  - Interoperability
  - Security
- **Reliability**
  - Maturity
  - Fault tolerance
  - Recoverability
- **Usability**
  - Understandability
  - Learnability
  - Operability
- **Efficiency**
  - Time behavior
  - Resource behavior
- **Maintainability**
  - Analyzability
  - Changeability
  - Stability
  - Testability
- **Portability**
  - Adaptability
  - Installability
  - Conformance
  - Replaceability

Fault vs Failure



Different Products Requires Different Quality

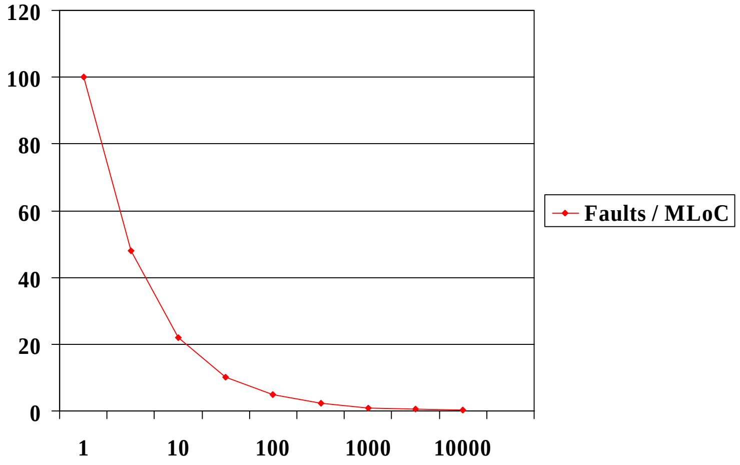
- Consumer products
- Professional tools
- Industrial systems
- Medical systems
- Auto pilots
- ….

The Impossible Equation

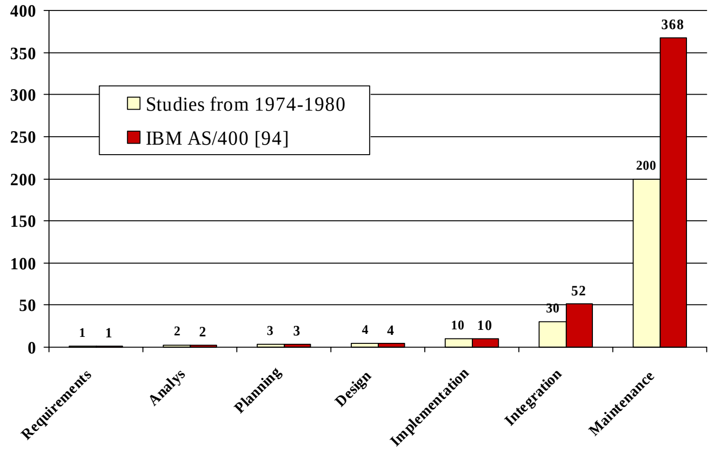
- Functionality
- Time
- Cost
- **Quality**

Quality is our top priority, as long as it is does not interfere with any other goals

Choosing the Quality Level

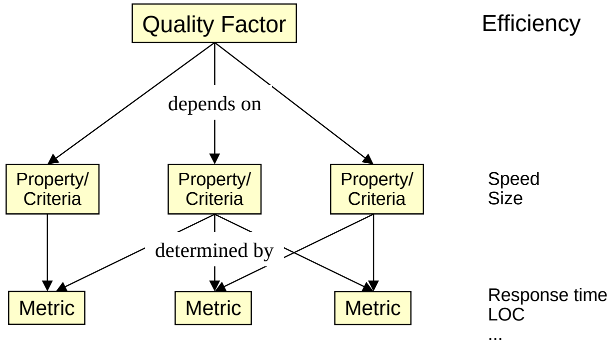


Relative Costs of an Error



See [Schach 97].

How To Measure Quality?



- Metrics are (objective) measurements to determine (subjective) quality factors

Quality Metrics

- Hard to find objective metrics.
- Metrics chosen from ease of measurement rather than from importance.
- Only successful in mature companies.*
- In practice, mainly subjective metrics are in use.

**) Ericsson is <u>not</u> one of them*

Some Example Metrics

- To measure efficiency
  - Time behaviour
    - Transactions per second
    - Response time
    - Screen refresh time
  - Resource behaviour
    - KBytes of executables
    - LOC
    - Number of processors
- To measure usability
  - Training time
  - Number of help frames
- To measure reliability
  - MTTF (Mean Time To Failure)
  - Availability
- To measure robustness
  - Time to restart after a failure
  - Probability of data corruption on failure
- To measure portability
  - Number of target systems
  - Percentage of target dependent statements

Purpose of Measurement

- Analysis: Determine current quality
- Prediction: Predict future quality
- Measurement possible on:
  - Produced output
    - Code
    - Documentation
    - Design
  - Processes
    - Construction phase
    - Test phase
  - Resources
    - Personnel
    - Budget

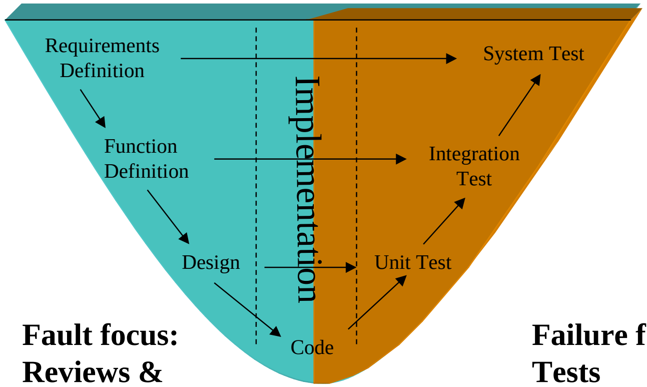
Achieving Quality

- Construction of Quality
  - *Quality is created along the road!!!*

- Reviews and Inspections
- Testing

} **Quality Assurance**

Quality Assurance in The V-Model



Requirements Definition

Function Definition

Design

Code

Implementation

Unit Test

Integration Test

System Test

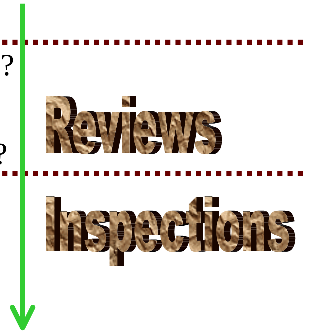
Fault focus: Reviews & Inspections

Failure focus: Tests

Reviews & Inspections

R&I: Formality Levels

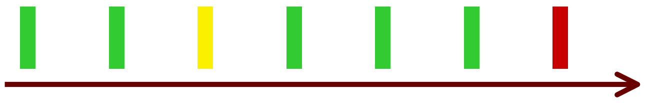
- Coffee talk: Trying ideas.
- Walkthrough: Better solution?
- 1/3 Presentation: On track?
- Frequent Review: Right solution?
- Inspection: Good enough?



Higher formality

The Authoring Process: R&I Strategy

- 🟩 Frequent Review
- 🟨 1/3 Presentation
- 🟥 Inspection



Frequent Review

- Informal, performed by peer designer
- Read, understand, question
- “Second opinion”

1/3 Presentation

- Stakeholder representatives
- Present the approach
- Early adjustment of direction

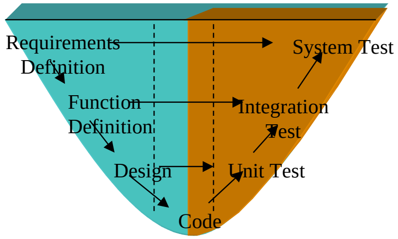
Inspection

- ◆ Formal
- ◆ Quality measurement
- ◆ Well defined process
- ◆ Requires defined standards
- ◆ Quality of metrics depend on quality of standards

Testing

Principle:

- ◆ Test design is done in parallel with design activities, using the same input data, but unaware of the technical solution.
- ◆ If coding and code testing is done by the same person: write code tests before coding.



How much testing is enough?

- ◆ It is never enough?
- ◆ When you’ve proved the system is correct?
- ◆ When you’ve done what you planned?
- ◆ When your customer is happy?
- ◆ When you’re confident the system works correctly?

It depends on the risks for your system!

Risks

- Loss of life?
- Loss of credibility?
- Disturbance in customers business?
- Missed market window?
- Unnecessary development cost?
- ...

Why not just test everything? Remember that…..

- … number of possible scenarios = 2^{i+e}
- … if transitions are non-atomic, the state may be altered by other use cases during the execution of this use case.
  Number of possible scenarios >> 2^{i+e}

Testing Steps

- Unit test (Basic test, Module test...)
- Integration test
- System test
  - Function test*
  - Performance test
- Acceptance test
- Installation test

*) Often considered as two separate activities, single function vs functions in complete system

Low Level Quality Assurance (1)

- Basic test (Dynamic testing)
  - Execution of code on lowest level
  - Automated tools
  - Test scripts
  - Test harnesses
  - A good test script gives you the courage to redesign!
- Desk check
  - Check list for common faults
  - Checking rate ~100 LoC / hour

Low Level Quality Assurance (2)

- Tool supported analysis (Static testing)
  - Execution coverage
  - Performance
  - Memory leaks
  - Common pitfalls
  - Complexity
  - Array bounds

Error Handling

- Highlight faults
  - Never hide a fault
  - Disastrous symptoms are good during testing
  - Use error logs for delivered systems
- Avoid failures
  - Try to reduce effect in target system.
  - Failure avoidance strategy depends on criticality
- Unusual conditions are not faults (e.g. disk full)
  - Lack of handling of them are!

Dynamic Test Approaches (1)

- Systematic
  - Black Box
    - Intended sequence / possible event
    - State / possible event
  - White Box
    - Weak spots
    - Coverage Driven
      - Statement
      - Decision
      - Condition
      - Multiple condition

Dynamic Test Approaches (2)

- Non systematic “Happy testing”
  - Ad hoc
  - Error guessing
  - User testing

Integration Testing

- Build
- Build + smoke test?
- Big bang
- Top-down
- Bottom up

System Test

- Functional
  - Requirements
  - Use Cases
- Non functional
  - Performance
  - Load
  - Recovery
  - Usability
  - Installation
  - …..

Function Test Approach Example, Use Case Based

- All Use Cases
  - All Scenarios
    - All Data
      - Equivalence partitioned
      - Boundary values
      - Invalid data
- Happy usage
  - Useful and reasonable combinations of use cases
  - Stressing
    - Pulling cables
    - Removing diskettes
    - …..

Regression Testing

- Rerunning existing tests after a change.
- Traceability necessary to identify test cases.
- The most expensive activity for a small change.
- Cost for regression testing discourages improvements in reused quality.
- Automated testing crucial!