Software Quality Engineering:

Testing, Quality Assurance, and

Quantifiable Improvement

Jeff Tian, tian@engr.smu.edu
www.engr.smu.edu/∼tian/SQEbook

Chapter 3. Quality Assurance (QA)

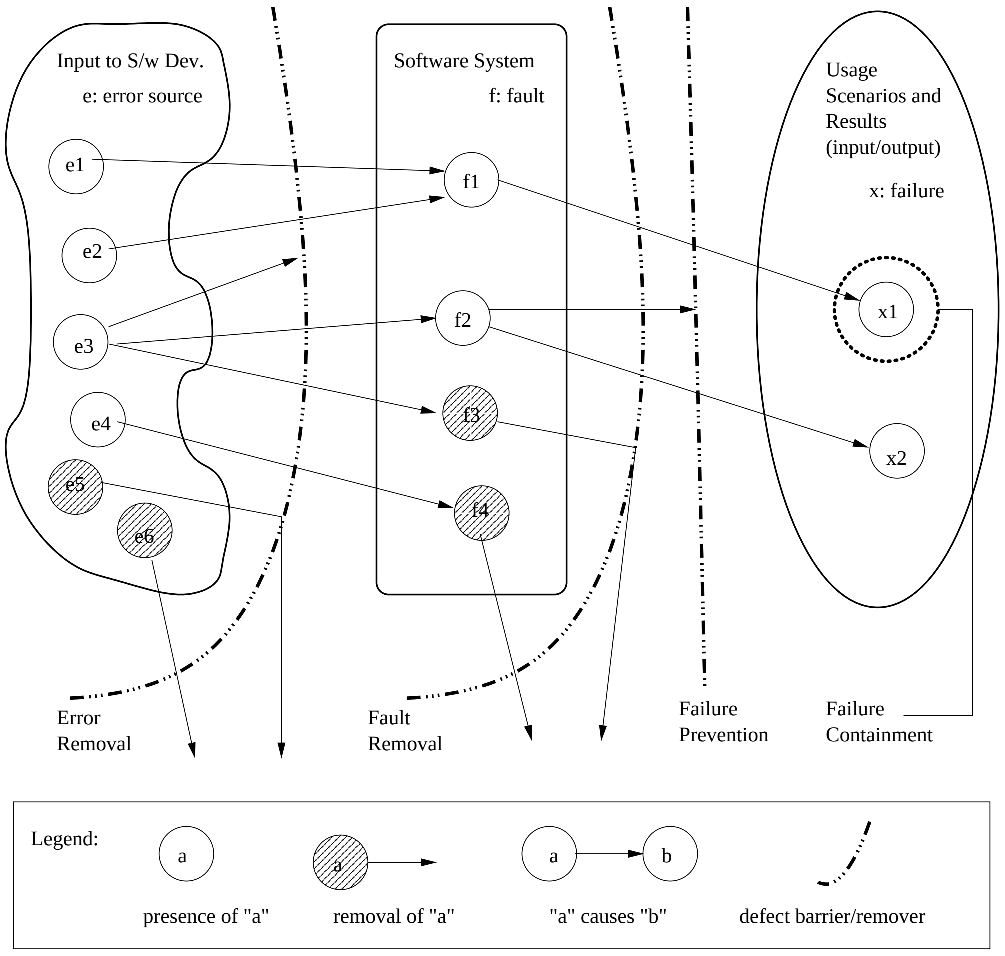
- QA as Dealing with Defect
- Defect Prevention
- Defect Detection and Removal
- Defect Containment

Defect vs. QA

- QA: quality assurance
  - ▷ focus on correctness aspect of Q
  - ▷ QA as dealing with defects
    - – post-release: impact on consumers
    - – pre-release: what producer can do
  - ▷ what: testing & many others
  - ▷ when: earlier ones desirable (lower cost) but may not be feasible
  - ▷ how ⇒ classification below

- How to deal with defects:
  - ▷ prevention
  - ▷ removal (detect them first)
  - ▷ containment

QA Classification



- Fig 3.1 above (p.30): QA as barriers
  - ▷ dealing with errors, faults, or failures
  - ▷ removing or blocking defect sources
  - ▷ preventing undesirable consequences

Error/Fault/Failure & QA

- Preventing fault injection
  - ▷ error blocking (errors $\not\Rightarrow$ faults)
  - ▷ error source removal

- Removal of faults (pre: detection)
  - ▷ inspection: faults discovered/removed
  - ▷ testing: failures trace back to faults

- Failure prevention and containment:
  - ▷ local failure $\not\Rightarrow$ global failure
    – via dynamic measures to tolerate faults
  - ▷ failure impact$\downarrow$ $\Rightarrow$ safety assurance

Defect Prevention Overview

- Error blocking
  - ▷ error: missing/incorrect actions
  - ▷ direct intervention to block errors
    $\Rightarrow$ fault injections prevented
  - ▷ rely on technology/tools/etc.

- Error source removal
  - ▷ root cause analysis
    $\Rightarrow$ identify error sources
  - ▷ removal through education/training/etc.

- Systematic defect prevention via process improvement.

- Details: Chapter 13.

Formal Method Overview

- Motivation
  - ▷ fault present:
    – revealed through testing/inspection/etc.
  - ▷ fault absent: formally verify.
    (formal methods $\Rightarrow$ fault absent)
- Basic ideas
  - ▷ behavior formally specified:
    – pre/post conditions, or
    – as mathematical functions.
  - ▷ verify "correctness":
    – intermediate states/steps,
    – axioms and compositional rules.
  - ▷ Approaches: axiomatic/functional/etc.
- Details: Chapter 15.

Inspection Overview

- Artifacts (code/design/test-cases/etc.) from req./design/coding/testing/etc. phases.

- Informal reviews:
  - ▷ self conducted reviews.
  - ▷ independent reviews.
  - ▷ orthogonality of views desirable.

- Formal inspections:
  - ▷ Fagan inspection and variations.
  - ▷ process and structure.
  - ▷ individual vs. group inspections.
  - ▷ what/how to check: techniques .

- Details: Chapter 14.

Testing Overview

- Product/Process characteristics:
  - ▷ object: product type, language, etc.
  - ▷ scale/order:
    unit, component, system, ...
  - ▷ who: self, independent, 3rd party
- What to check:
  - ▷ verification vs. validation
  - ▷ external specifications (black-box)
  - ▷ internal implementation (white/clear-box)
- Criteria: when to stop?
  - ▷ coverage of specs/structures.
  - ▷ reliability $\Rightarrow$ usage-based testing
- Much, much more in Part II.

Fault Tolerance Overview

- Motivation
  - ▷ fault present but removal infeasible/impractical
  - ▷ fault tolerance ⇒ contain defects
- FT techniques: break fault-failure link
  - ▷ recovery: rollback and redo
  - ▷ NVP: N-version programming
    - – fault blocked/out-voted
- Details: Chapter 16.

Safety Assurance Overview

- Extending FT idea for safety:
  – fault tolerance to failure "tolerance"

- Safety related concepts:
  - ▷ safety: accident free
  - ▷ accident: failure w/ severe consequences
  - ▷ hazard: precondition to accident

- Safety assurance:
  - ▷ hazard analysis
  - ▷ hazard elimination/reduction/control
  - ▷ damage control

- Details: Chapter 16.