Software Quality

Lectire 1

Outline

- Overview
- What is software quality?
  - Perspectives and Expectations
  - Quality Frameworks and ISO-9126
  - Correctness, Defect, and Quality
  - A Historical Perspective
- Quality Assurance
  - QA as Dealing with Defect
  - Defect Prevention
  - Defect Detection and Removal
  - Defect Containment
- Conclusion

General expectations

- General expectation:
  - “good" software quality

- Objects of our study: software
  - software products, systems, and services
  - stand-alone to embedded
  - software-intensive systems
  - wide variety, but focus on software

Quality Expectations

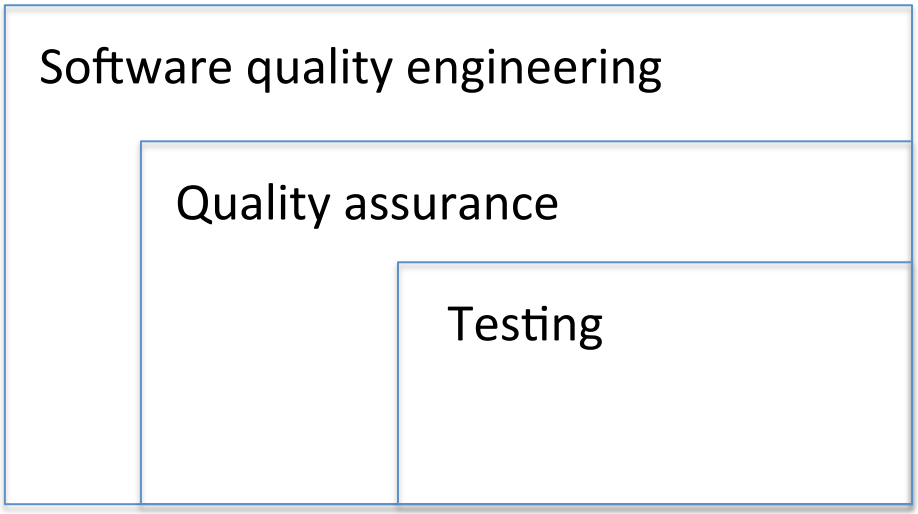
- People: Consumers vs producers
  - quality expectations by consumers
  - to be satisfied by producers through software quality engineering (SQE)
- Deliver software system that
  - does what it is supposed to do
    - ✓ needs to be “validated”
  - does the things correctly
    - ✓ needs to be “verified”
  - show/demonstrate/prove it (“does")
    - ✓ modeling/analysis needed

Meeting Quality Expectations

- Difficulties in achieving good quality:
  - size: MLOC products common
  - complexity
  - environmental stress/constraints
  - flexibility/adaptability expected
- Other difficulties/factors:
  - product type
  - cost and market conditions
  - .....
- “no silver bullet", but...
  - SQE (software quality engineering) helps

SQE as an Answer

- Major SQE activities:
  - Testing: remove defect & ensure quality
  - Other QA alternatives to testing
  - How do you know: analysis & modeling
- Scope and content hierarchy

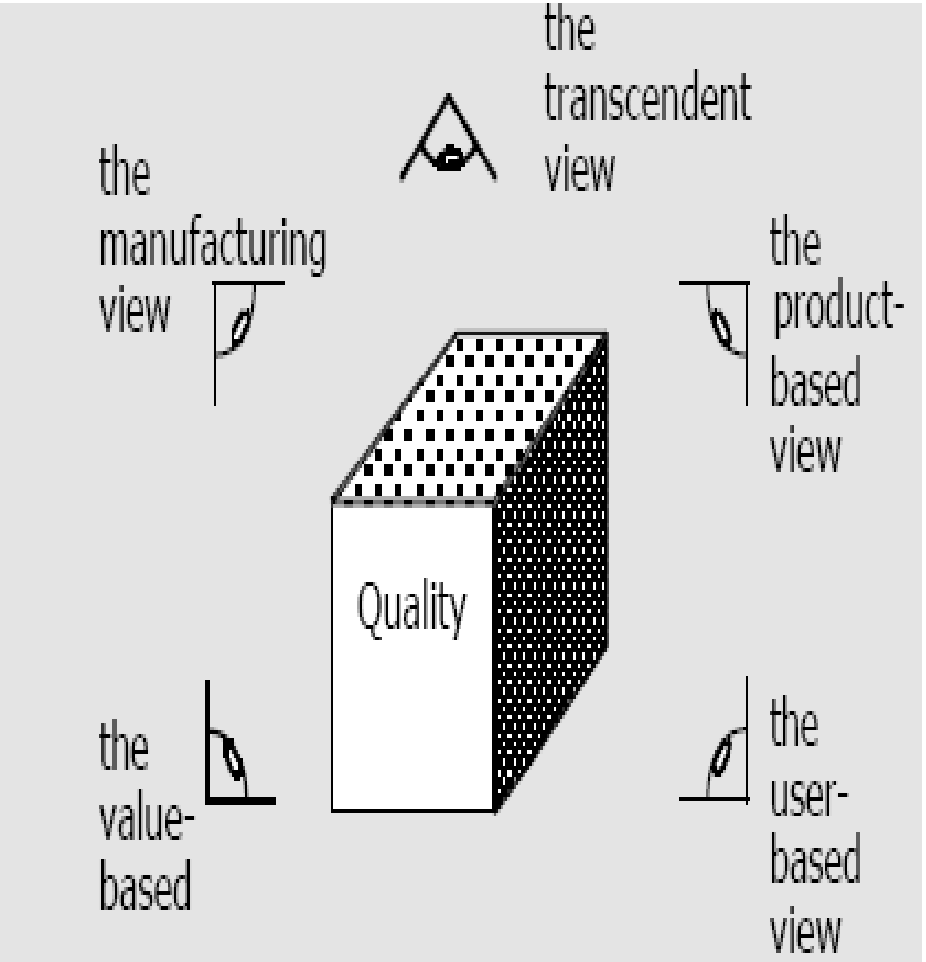


Outline

- Overview
- **What is software quality?**
  - **Perspectives and Expectations**
  - **Quality Frameworks and ISO-9126**
  - **Correctness, Defect, and Quality**
  - **A Historical Perspective**
- Quality Assurance
  - QA as Dealing with Defect
  - Defect Prevention
  - Defect Detection and Removal
  - Defect Containment
- Conclusion

Perspectives and Expectations

- General: “good" software quality
- Perspectives:
  - people/subject's view, software as object
- Expectations: quality characteristics & level
- In Kitchenham & Pfleeger (1996, Pfleeger 2002):
  - Transcendental view: seen/not-defined.
  - User view: fitness for purpose.
  - Manufacturing view: conform to specification
  - Product view: inherent characteristics
  - Value-based view: willing to pay.



Quality Perspectives

- Perspectives: subject and object
- Subject: people's perspectives
  - external/**consumer**: customers and users
  - internal/**producer**: developers, testers, and managers
  - other: 3rd party participants, etc.
  - users generalized: other systems etc.
  - focus on external/consumer side
- Objects of our study:
  - software products, systems, and services
  - stand-alone, embedded, etc.
  - affect quality definitions/expectations

Quality Expectations

- Expectations from different people
- External/consumer expectations:
  - “good enough" for the price
    - ✓ fit-for-use, doing the “right things” – performs right functions as specified
    - ✓ conformance, doing “things right” - performs these specified functions correctly over repeated use or over a long period of time

    ➔ validation and verification (V&V)
  - customer vs user (+price?)
  - generalized user: other hw/sw/system/etc.
- Expectations for different software:
  - general: functionality & reliability,
  - usability: GUI/end-user/web/etc.,
  - interoperability: embedded systems,
  - safety: safety-critical systems, etc.

Quality Expectations 2

- Internal/producer:
  - “good enough" for the cost
    - ✓ mirror consumer side
    - ✓ functionality & correctness via V&V
  - managers: software process, standards, languages, tools
  - service related: maintainability, usability, modifiability
  - interfacing units: interoperability
  - 3rd party: modularity, portability
- Different expectations for different types of products and market segments too.
- Different QA/SQE activities needed.

ISO-9126 Quality Framework

- Various models or frameworks have been proposed to accommodate the different quality views and expectations, and to define quality and related attributes, features, characteristics, and measurements.
- ISO-9 126 (ISO, 2001) provides a hierarchical framework for quality definition, organized into quality characteristics and sub-characteristics
- ISO 9126 quality characteristics:
  - Functionality: what is needed?
  - Reliability: function correctly.
  - Usability: effort to use.
  - Efficiency: resource needed.
  - Maintainability: correct/improve/adapt.
  - Portability: one environment to another.

ISO-9126 Quality Framework 2

- Functionality - existence of a set of functions and their specified properties
  - Suitability
  - Accuracy
  - Interoperability
  - Security
- Reliability - capability of software to maintain its level of performance under stated conditions for a stated period of time
  - Maturity
  - Fault tolerance
  - Recoverability
- Usability - effort needed for use, and the individual assessment of such use, by a stated or implied set of users
  - Understandability
  - Learnability
  - Operability

ISO-9126 Quality Framework 3

- Efficiency - relationship between the level of performance of the software and the amount of resources used, under stated conditions
  - Time behavior
  - Resource behavior
- Maintainability - the effort needed to make specified modifications.
  - Analyzability
  - Changeability
  - Stability
  - Testability
- Portability - the ability of software to be transferred from one environment to another
  - Adaptability
  - Installability
  - Conformance
  - Replaceability

ISO-9126 Quality Framework 4

- Impact and limitations:
  - Characteristics into sub-characteristics
  - Comprehensive framework
  - Strict hierarchy ➔ other alternatives

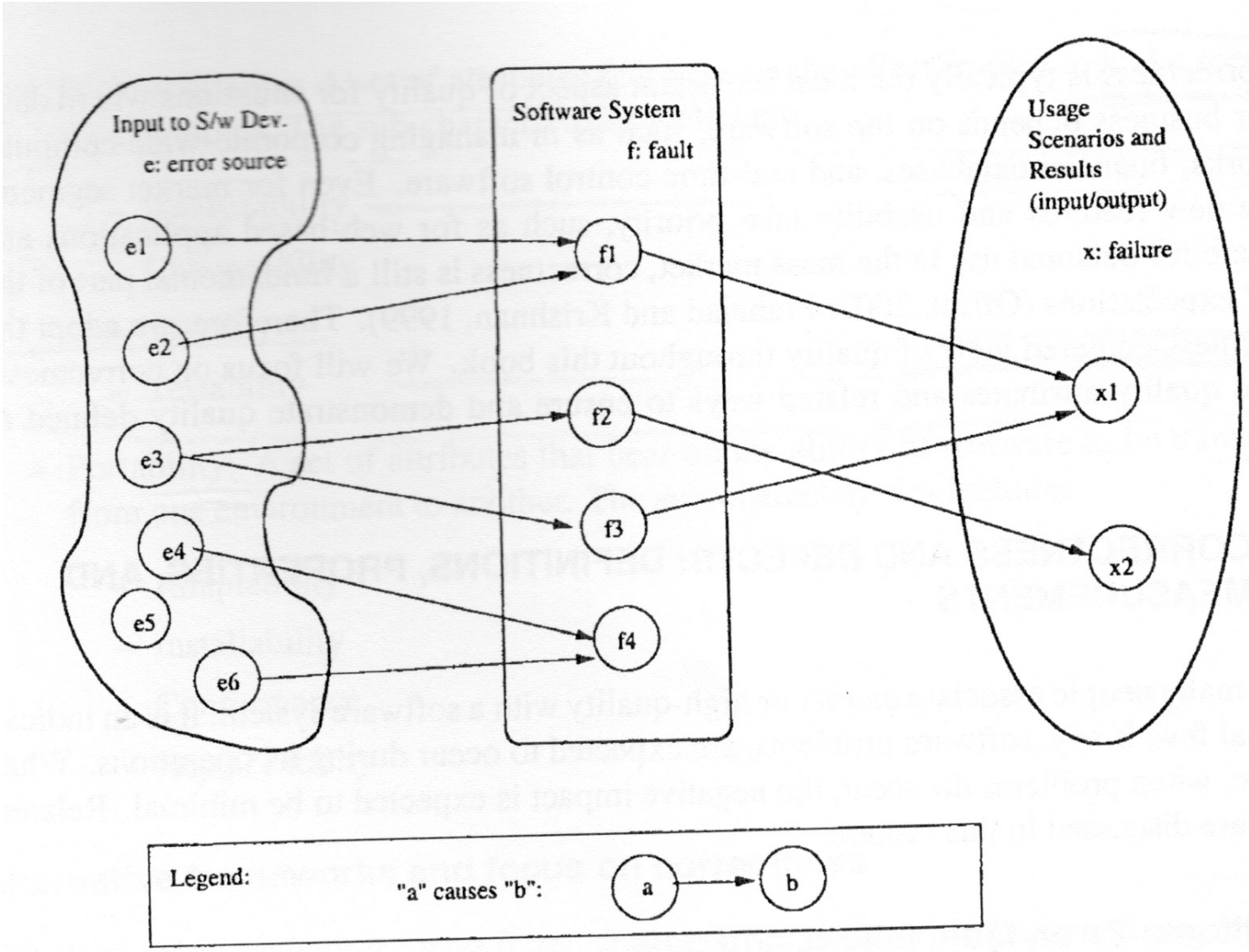
Other Quality Frameworks

- Adaptation of ISO-9126:
  - customized for companies
    - ✓ e.g., IBM's CUPRIMDSO (capability, usability, performance, reliability, installation, maintenance, documentation, and service).
  - adapted to application domains
    - ✓ Primary (reliability, usability, security) and secondary (availability, scalability, maintainability, and time to market) for Web
- Other quality frameworks/mega-models
- McCall: factors, criteria, and metrics
- Basili: GQM (goal-question-metric)
- SEI/CMM: process focus/levels
- Dromey: component reflects Q-attributes
- Defect-based view: common in industry
  - cost of defect: by Boehm, NIST, etc.

Correctness, Defect and Quality

- High quality ≈ low defect
  - intuitive notion related to correctness
  - quality problem ≈ defect impact
  - widely accepted, but need better definitions
- Defect/bug definition
  - failure: external behavior
    - ✓ deviation from expected behavior
  - fault: internal characteristics
    - ✓ cause for failures
  - error: incorrect/missing human action
    - ✓ error source: conceptual mistakes etc.
  - defect: error, fault, failure collectively
  - bug/debug: problematic terms, avoid

Correctness, Defect and Quality 2



Correctness, Defect and Quality 3

- Middle box - the software system as represented by its artifacts - software code, designs, specifications, requirement documents, etc..
- Left box - the input to the software development activities include conceptual models and information, developers with certain knowledge and experience, reusable software components, etc.
- The *errors* as missing or incorrect human actions are not directly depicted within one box, but rather as actions leading to the injection of faults in the middle box because of some error sources in the left box.
- Right box - Usage scenarios and execution results describe the input to software execution, its expected dynamic behavior and output, and the overall results. A subset of these behavior patterns or results can be classified as failures when they deviate from the expected behavior.

Defining Quality in SQE

- Quality: views and attributes

View	Attribute	
	Correctness	Other
Customer (external)	Failures: Reliability Safety etc.	Maintainability Readability Portability Performance Installability Usability, etc.
Developer (internal)	Faults: Count Distribution Density etc.	Design Size Change Complexity presentation control data, etc.

- SQE focus: correctness-related

Quality: Historical Perspective

- Software vs other products/systems:
  - pre-software/IT: manufacturing process
    - ✓ physical-object attributes (defects)
  - service: manage expectations:
    - ✓ 0 defect -- 0 defection
  - IT and software: below
- The new meaning of quality in the information age (Prahalad & Krishnan 1999):
  - Conformance/adaptability/innovation
  - Traditional: conformance only
  - Domain specific (for info. age):
    - ✓ specificity, stability, evolvability

Quality: Historical Perspective 2

- A historical perspective of SE, in 4 stages (Musa & Everett, 1990):
  - functional: focus on automation
  - schedule: timely/orderly product intro
  - cost: competitive marketplace
  - reliability: meet user expectations

Quality Management Philosophies

- Quality according to Crosby
- Quality according to Deming
- Quality according to Feigenbaum
- Quality according to Ishikawa
- Quality according to Juran
- Quality according to Shewhart

Quality according to Crosby

- The first erroneous assumption is that quality means goodness, or luxury or shininess. The word “quality” is often used to signify the relative worth of something in such phrases as “good quality”, “bad quality” and “quality of life” - which means different things to each and every person. As follows quality must be defined as “conformance to requirements” if we are to manage it. Consequently, the nonconformance detected is the absence of quality, quality problems become nonconformance problems, and quality becomes definable
  - Quality is defined as conformance to requirements, not as “goodness” or “elegance”
  - The system for causing quality is prevention
  - The performance standard must be Zero Defects, not "that's close enough"
  - The measurement of quality is the cost of quality

Quality according to Deming

- The problem inherent in attempts to define the quality of a product, almost any product, where stated by the master Walter A. Shewhart. The difficulty in defining quality is to translate future needs of the user into measurable characteristics, so that a product can be designed and turned out to give satisfaction at a price that the user will pay. This is not easy, and as soon as one feels fairly successful in the endeavor, he finds that the needs of the consumer have changed, competitors have moved in etc.

Quality according to Feigenbaum

- Quality is a customer determination, not an engineer's determination, not a marketing determination, nor a general management determination. It is based on upon the customer's actual experience with the product or service, measured against his or her requirements – stated or unstated, conscious or merely sensed, technically operational or entirely subjective – and always representing a moving target in a competitive market.

Product and service quality can be defined as: The total composite product and service characteristics of marketing, engineering, manufacture and maintenance though witch the product and service in use will meet the expectations of the customer.

- Quality must be defined in terms of customer satisfaction, that quality is multidimensional (it must be comprehensively defined), and as the needs are changing quality is a dynamic concept in constant change as well.

Quality according to Ishikawa

- We engage in quality control in order to manufacture products with the quality which can satisfy the requirements of consumers. The mere fact of meeting national standards or specifications is not the answer, it is simply insufficient. International standards established by the International Organization for Standardization (ISO) or the International Electrotechnical Commission (IEC) are not perfect. They contain many shortcomings. Consumers may not be satisfied with a product which meets these standards. We must also keep in mind that consumer requirements change from year to year and even frequently updated standards cannot keep the pace with consumer requirements. How one interprets the term "quality" is important. Narrowly interpreted, quality means quality of products. Broadly interpreted, quality means quality of product, service, information, processes, people, systems etc. etc.

Quality according to Juran

- The word quality has multiple meanings. Two of those meanings dominate the use of the word: 1) Quality consists of those product features which meet the need of customers and thereby provide product satisfaction. 2) Quality consists of freedom from deficiencies. Nevertheless, in a handbook such as this it is most convenient to standardize on a short definition of the word quality as “fitness for use”
  - **Quality planning**: A process that identifies the customers, their requirements, the product and service features that customers expect, and the processes that will deliver those products and services with the correct attributes and then facilitates the transfer of this knowledge to the producing arm of the organization.
  - **Quality control**: A process in which the product is examined and evaluated against the original requirements expressed by the customer. Problems detected are then corrected.
  - **Quality improvement**: A process in which the sustaining mechanisms are put in place so that quality can be achieved on a continuous basis. This includes allocating resources, assigning people to pursue quality projects, training those involved in pursuing projects, and in general establishing a permanent structure to pursue quality and maintain the gains secured.

Quality according to Shewhart

- There are two common aspects of quality: One of them has to do with the consideration of the quality of a thing as an objective reality independent of the existence of man. The other has to do with what we think, feel or sense as a result of the objective reality. In other word, there is a subjective side of quality.

Software quality

- So, what is software quality?
  - many aspects/perspective, but

    correctness-centered in SQE

Outline

- Overview
- What is software quality?
  - Perspectives and Expectations
  - Quality Frameworks and ISO-9126
  - Correctness, Defect, and Quality
  - A Historical Perspective
- **Quality Assurance**
  - **QA as Dealing with Defect**
  - **Defect Prevention**
  - **Defect Detection and Removal**
  - **Defect Containment**
- Conclusion

QA

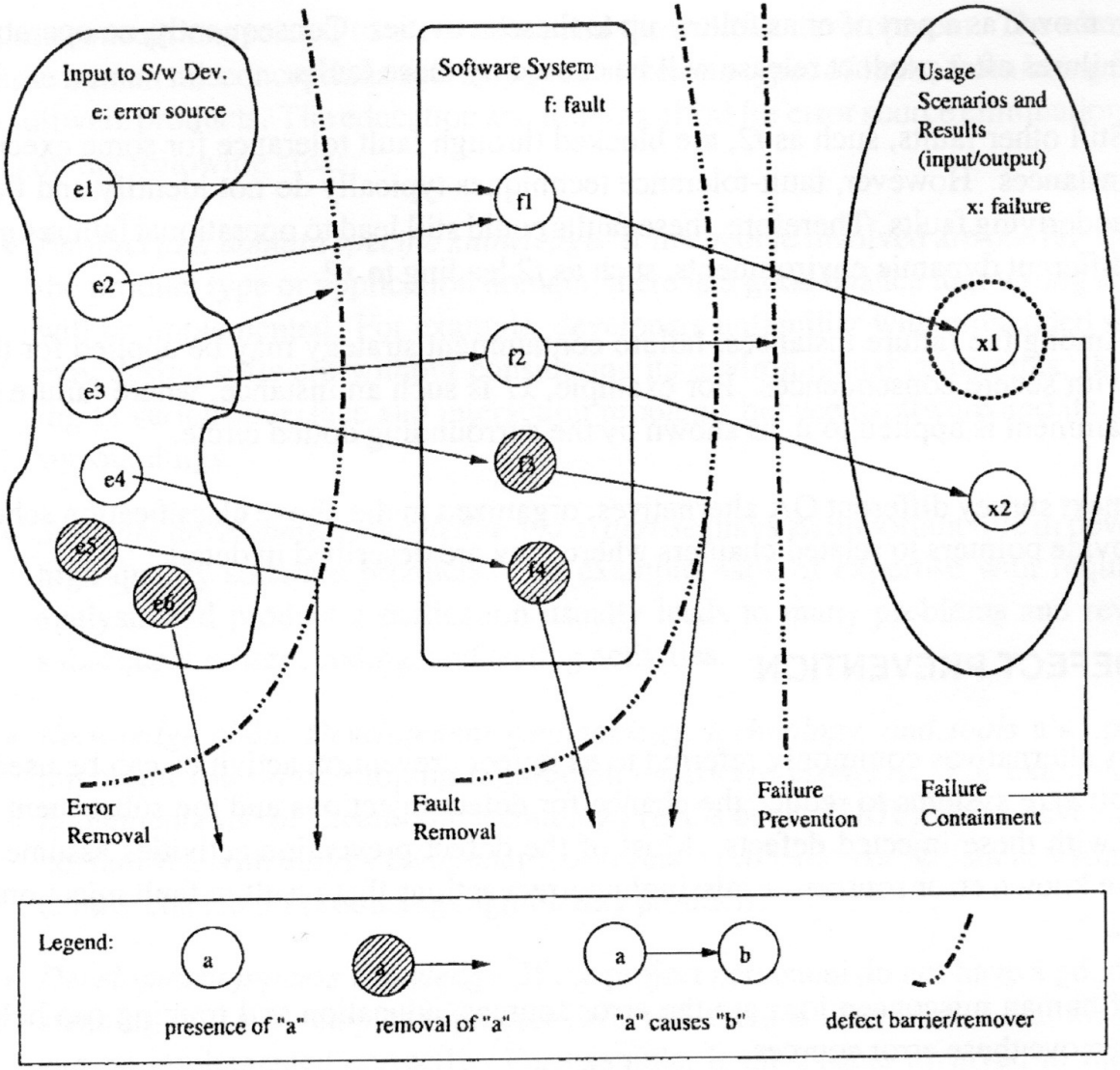
- The central activities for quality assurance (QA) can be viewed as to ensure that few, if any, defects remain in the software system when it is delivered to its customers or released to the market. Furthermore, we want to ensure that these remaining defects will cause minimal disruptions or damages

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 Classication



QA Classication 2

- 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 ≠ 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 ≠ global failure
    - ✓ via dynamic measures to tolerate faults
  - failure impact ↓ → safety assurance

Defect Prevention Overview

- Error blocking
  - error: missing/incorrect actions
  - direct intervention to block errors

    ➔fault injections prevented
  - rely on technology/tools/etc.
- Error source removal
  - If human misconceptions are the error sources, education and training can help us remove these error sources.
  - if imprecise designs and implementations that deviate from product specifications or design intentions are the causes for faults, formal methods can help us prevent such deviations.
  - if non-conformance to selected processes or standards is the problem that leads to fault injections, then process conformance or standard enforcement can help use prevent the injection of related faults.
  - If certain tools or technologies can reduce fault injections under similar environments, they should be adopted.
  - root cause analysis

    ➔identify error sources
  - removal through education/training/etc.
- Systematic defect prevention via process improvement.

Formal Method Overview

- Motivation
  - fault present:
    - ✓ revealed through testing/inspection/etc.
  - fault absent: formally verify.

  (formal methods ➔ 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.

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

Inspection Overview 2

- Inspections are **critical reading and analysis** of software code or other software artifacts, such as designs, product specifications, test plans, etc.
- ***I***nspections are typically conducted by multiple human inspectors, through some coordination process. Multiple inspection phases or sessions might be used.
- Faults are detected directly in inspection by human inspectors, either during their individual inspections or various types of group sessions.
- Identified faults need to be removed as a result of the inspection process, and their removal also needs to be verified.
- The inspection processes vary, but typically include some planning and follow-up activities in addition to the core inspection activity.
- The formality and structure of inspections may vary, from very informal reviews and walkthroughs, to fairly formal variations of Fagan inspection, to correctness inspections approaching the rigor and formality of formal methods.

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 ➔ usage-based testing

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

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

Conclusion

- QA alternatives
  - *Defect prevention* through error source elimination and error blocking activities, such as education and training, formal specification and verification, and proper selection and application of appropriate technologies, tools, processes, or standards..
  - *Defect reduction* through inspection, testing, and other static analyses or dynamic activities, to detect and remove faults from software.
  - *Defect containment* through fault tolerance, failure prevention, or failure impact minimization, to assure software reliability and safety.


Q&A