Quality Assurance

Course Code: CSC4133 Course Title: Software Quality and Testing

Dept. of Computer Science
Faculty of Science and Technology

Lecturer No:	**5**	**Week No:**	**3**	**Semester:**	**Spring 19_20**
Lecturer:	*Prof. Dr. kamruddin Nur,* *kamruddin@aiub.edu*				

Lecture Outline

- QA Activities
  - Defect Prevention
  - Defect Reduction
  - Defect Containment

Objectives and Outcomes

- **Objectives**: To understand the major activities of quality assurance as it deals with defects, to understand the activities of defect prevention, defect reduction and defect containment.

- **Outcomes**: Students are expected to be able to explain the activities of defect prevention, defect reduction and defect containment.

Defect vs. QA

QA ==> Quality Assurance

- Focus on correctness aspect of quality
- QA as dealing with defects
  - **Post-release**: Impact on consumers
  - **Pre-release**: What producer can do

- **What?** Testing & many other activities
- **When?** Earlier ones desirable (lower cost), but may not be feasible
- **How?** Classification next slide

Classification of QA activities

- **What are the QA activities to deal with defects?**

  QA activities can be classified into three generic categories –

  1) **Defect Prevention**
  2) **Defect Reduction**
  3) **Defect Containment**

Error/Fault/Failure & QA Activities

- **Preventing** fault injection
  - Error blocking (error ==> faults)
  - Error source removal
- **Removal** of faults (pre: detection)
  - **Inspection** : Fault 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

[1] Defect prevention

- **Defect prevention** can be done in two generic ways-
  a) Error source removal
  b) Error blocking
  – These QA activities prevent certain types of faults from being injected into the software
  – Since errors are the missing/incorrect human actions that lead to injection of **faults** into software systems, we can *directly correct or block these actions, or remove the underlying causes for them*

[1] Defect prevention

a) Error source removal : Eliminating certain error sources such as eliminating ambiguities or correcting human misconceptions, which are the root causes for errors.
   - Root cause analysis ➔ identify error sources
   - Remove through education/training/etc.

[1] Defect prevention

b) Error blocking: Fault prevention or blocking by directly correcting or blocking these missing or incorrect human actions.

- **Error** : missing/incorrect human actions
- Direct intervention to block errors ➔fault injections prevented

- Systematic defect prevention via process improvements
- Defect prevention activities can be used for most software systems to reduce the chance for defect injections and the subsequent cost to deal with these injected defects.

[1] Defect prevention

- Most defect prevention activities assume that there are known error sources or missing/incorrect actions that result in fault injections, as follows:
  - 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 intensions are the causes for faults, **formal methods** can help prevent such deviations
  - If **non-conformance to selected processes or standards** is the problem that leads to fault injections, then **process conformance/standard enforcement** can help prevent the injection of related faults
  - If **certain tools/technologies** can reduce fault injections under similar environments, they **should be adopted**

[1] Defect prevention

- Root cause analyses are needed to establish these pre-conditions, or *root causes* for injected or potential faults, so that appropriate defect prevention activities can be applied to prevent injection of similar faults in the future. Once such causal relations are established, appropriate QA activities can then be selected & applied for defect prevention.

  i. **Education and training**➔ provide people-based solutions for error source elimination.

  ii. **Formal method**➔ provide a way to eliminate certain error sources and to verify the absence of related faults.

  iii. **Other defect prevention techniques**

  - Analysis & modeling for defect prevention
  - Technologies, standards, and methodologies for defect prevention
  - Software tools to block defect injection

[2] Defect Reduction

- ***Defect Reduction through fault detection and removal*.** These QA alternatives detect and remove certain faults once they have been injected into the software systems.
- **Most traditional QA activities fall into this category (defect reduction)**
  - **Inspection** directly detects and removes faults from the software code, design etc.
  - **Testing** removes faults based on related failure observations during program execution
- Various other means, based on either static analyses or observations of dynamic executions, can be applied to reduce the # of faults in a software system.

[2] Defect Reduction

- For most large software systems in use today, it is unrealistic to expect the defect prevention activities surveyed above to be 100% effective in preventing accidental fault injections.
- Need effective techniques to remove as many of the injected faults as possible under project constraints.
  - Inspection
  - Testing
  - Other techniques & risk identification

[2] Defect Reduction

- **Inspection**: Direct fault detection & removal
  - Inspection is the most commonly used **static technique** for defect detection and removal
  - Inspections are critical reading & analysis of software code or other software artifacts, such as designs, product specifications, test plans etc.
  - Inspection are typically conducted by multiple human inspectors, through some coordination process. Faults are detected directly in inspection by inspectors
  - ***Formality*** and ***structures*** of Inspections **vary**
  - **Informal reviews ==> Walkthroughs**
  - **Formal review ==> Inspections**
  - Inspection is most commonly applied to code, but it could be used throughout the development process (particularly early in s/w dev.)

[2] Defect Reduction

- **Testing**: Failure observation & fault removal
  - One of the most important activities of QA
  - Most commonly performed QA activity
  - Involves the execution of software and the observation of the program behavior/outcome. If a failure is observed, the execution record is then analyzed to locate & fix the fault(s) that caused the failure
  - *When can a specific testing activity be performed & related faults be detected?*
  - *What to test, and what kind of faults are found?*
  - *When, or at what defect level , to stop testing?*

[2] Defect Reduction

- Other techniques & risk identification:

  Besides Inspection & Testing, there are other static analyses and dynamic activities.

  - Various other static techniques – boundary value analysis, control flow and data flow analyses
  - Various other dynamic, execution-based techniques also available –symbolic execution, simulation, prototyping
  - They can help us detect and remove various defects early in the software development process, before large-scale testing becomes a viable alternative

[3] Defect Containment

- **Defect Containment** *through fault tolerance, failure prevention, or failure impact minimization, to assure software reliability and safety.*
  - Defect reduction activities can only reduce the number of faults to a fairly low level, but not completely eliminate them (because of the large size & high complexity of most software systems in use today)
  - The containment measures focus on the failures by either containing them to local areas so that there are no global failures observable to users, or limiting the damage caused by software system failures
  - Local failure ~~==~~> Global failure

[3] Defect Containment

- Defect Containment can be done in two generic ways:
  - Some QA alternatives, such as the use of **fault-tolerance** techniques, break the causal relation between faults and failures so that local faults will not cause global failures, thus "*tolerating*" these local faults.
  - A related extension to fault-tolerance is **containment** measures to avoid catastrophic consequences, such as death, personal injury, and severe property or environmental damages, in case of failures.

[3] Defect Containment

- Software fault-tolerance
  - Motivation
    - Fault present but removal infeasible/impractical
    - Fault tolerance ==> contain defects
  - FT techniques: break fault-failure link
    - Recovery: rollback & redo
    - **NVP (N-Version programming)** ==> Fault blocked

[3] Defect Containment

- Safety Assurance & Failure Containment
  - Extending FT idea for safety. Fault tolerance to failure "tolerance"
  - Safety : Accident free
  - Accident: Failure with severe consequences
  - Hazard: Pre-condition to accident
  - Safety Assurance: Hazard analysis, hazard elimination/reduction/control, damage control

Defect prevention, reduction & containment

- Existing software quality literature generally covers defect reduction techniques such as **testing** & **inspection** in more details than defect prevention activities, while largely ignore the role of defect containment in QA.

Books

1. *Software Quality Engineering: Testing, Quality Assurance and Quantifiable Improvement*, by Jeff Tian,

References

1. *Software Testing and Quality Assurance: Theory and Practice*, by Kshirasagar Naik, Priyadarshi Tripathy
2. *Software Quality Assurance: From Theory to Implementation*, by Daniel Galin
3. *Software Testing and Continuous Quality Improvement*, by William E. Lewis
4. *The Art of Software Testing*, by Glenford J. Myers, Corey Sandler and Tom Badgett
5. *Software Testing Fundamentals: Methods and Metrics* by Marnie L. Hutcheson