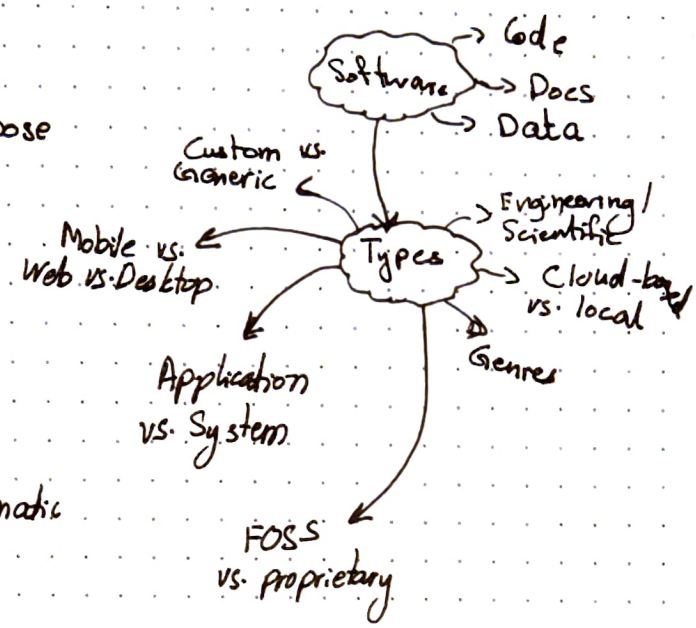
SQE L01 21-08-2024.

"Garbage in, garbage out."

Quality → fitness for purpose



Engineering: finding efficient solutions ~~about~~ for real-life problems in a systematic and efficient manner.

Problems compound (snowball effect, domino effect).

SQE L02 23-08-2024

ENIAC → Softwares → First CS dept. → Software Engineering
1940s late 1940s # in US 1962

THRAC-25
mid-late 1980s.

Denver, Colorado Airport Catastrophe

Washington prison malfunction: over 3000 prisoners released earlier

Toyota recalled 600,000 'Prius' cars in 2016

Nissan recalled ~3.5m

Flex

WHY SQE is a separate field?

- nature (virtual/logical vs. physical)
- size
- team setting
- formal ideation
- interfaces
- personnel turnover

Algorithmic Morality

SQE L03 28-08-2024

Software Quality Assurance Engineering

Great Bookstores
Regal Chowk, Mall Rd.
- Multi-line books
- Allied Books (Nagar ~~Book~~ Market)
- Pak Book Corporation (Queen's Rd.)
- Vanguard Books (Montgomery Rd.)

Quality:
- fitness for purpose
- conformance to explicit and implicit technical and managerial requirements

Perception of Quality ≠ Quality

Functional → services offered
Non-functional → constraints
managerial → budget (cost), schedule (time)

Software Quality Problems:

Software Supplier:
- Error: 'to err is human' incorrect action or inaction.
- Fault: a state of the software. not all errors become faults.
  → Bug / Defect

Software Customer:
- Failure: an event. not all faults become failures.
- Incident: detected ~~incidents~~ failure not all failures get noticed.

Software Quality Assurance: the set of systematic, planned activities required to provide adequate confidence in that the software produced through the software development process conforms to the explicit and implicit, technical and managerial requirements.
→ umbrella activity, process oriented

"A defect is a delivered fault."
- Beizman not for us.

"An accident is a severe failure."

→ subset of SQA, product oriented
Software Quality Control:

SQE L04 30-08-2024

Capers Jones → Software Project Management

"Short-cutting a day of QA activity early in the project is likely to add 3 to 10 days of unnecessary activity downstream."
- Capers Jones

- The most important deliverable of Reqs. Engineering:

Quality Factors → dimensions, attributes, content groups, SRS headings

-ilities

McCall's Quality Factor Model (Classic, 1977) → An abstraction of the real thing.

4 total 11 Quality Factors

product operation (5) (day-to-day operations)	product revision (3) (change)	product transition (3) (outside-world interaction)
- Correctness	- Maintainability	- Portability
- Reliability	- Flexibility	- Reusability
- Integrity	- Testability	- Interoperability
- Efficiency		
- Usability		

Correctness: functional requirements, output-mission, output accuracy, output completeness, output up-to-date-ness, availability (response time), standards

Reliability: failure rates, probabilities (critical systems)

Integrity: security (Authentication & Authorization)

Efficiency: optimal use of hardware resources (CPU, RAM, storage, communication bandwidth, battery life)

Usability: (ease-of-use)
- operations
- training

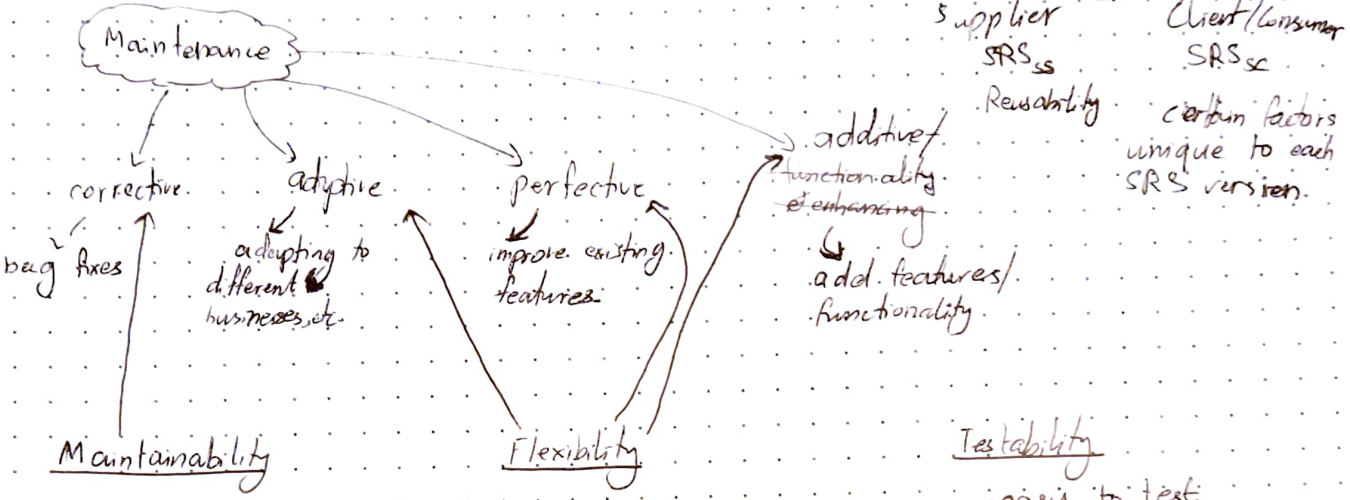
Question 03 (b)

SQE LO5 04-09-2024

BIOS does POST
↓
Power-On Self Test

McCall's QF Model

Product Revisions: change and maintenance

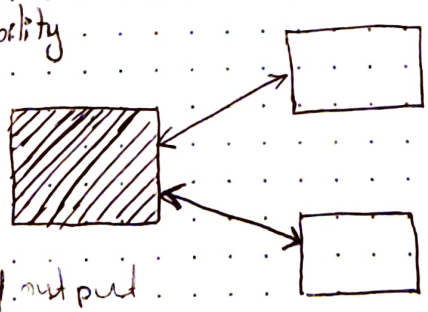


Software Supplier — Software Client/Consumer
SRS_{SS} — SRS_{SC}
Reusability — certain factors unique to each SRS version.

Testability
- easy to test
- intermediate results or logs
- allow operators and maintainers to test and diagnose the software.
- both comments and readability are important
- comments are not an excuse for bad programming practices.

Product Transition:
- Portability: cross-platform support.
- Reusability
  - for
  - with
  - improve documentation
  - increase cohesion
  - decrease coupling

- Interoperability



- standard output formats

Alternative QF Models.
- Evans & Marciniak (1987)
- Deutsch & Willis (1988)
  - Verifiability → mathematical verifications
  - Expandability ≈ Flexibility
  - Safety
  - Survivability ≈ Reliability
  - Manageability → administrative tools for change management.

human safety → removing unsafe condition

SQE LOG, 06-09-2024

Project Managers, UI/UX, programmers, testers, maintainers, dev ops, etc.
↳ many people can make mistakes

Components of SQA system

SQA applies to maintenance as well

Component Categories
- Pre-project
- Life-cycle
- Infrastructure
- Management
- Standards
- Human

Applicable to projects at hand

Applicable organization-wide

Pre-Project Components

-> Contract Review
- RFP -> Request for Proposal
- Draft -> Draft Review
- SC evaluates submissions
  - ↳ one gets a contract.
  - ↳ contract draft -> review

-> development & quality plans
- -> cost, resource, estimations
- -> WBS, Risk plans

SS ——— SC
↳ contract

Life-cycle components

Problem Analysis Phase | Solution Phase

Inc | RE | A | D | J | I | D/D | O-M | d/R

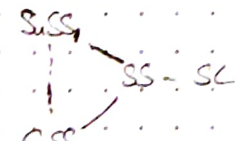
Part 1 | Part 2

-> Reviews
- team-based meetings
- variable degree of formality
- types: formal technical reviews (most formal), Inspections, Walk Throughs & (least formal)
  - ↳ FTR, Design Reviews, FDR
  - ↳ done by senior people

Peer Reviews

-> Testing
- regression, smoke, automated, manual, unit, integration, other tests.
  - black box, white box.

-> Expert Opinions
- outside reviews, etc
- Third Party



COTS / ROTS, GOTS
Commercial off-the-shelf components.

Question 03 (b)

SQE 106 (Continued)

Infrastructure Components

-> Procedures & Work Instructions
-> Supporting Quality Devices
  - Templates (SRS, project plan, risk plan, etc.)
  -
-> Training & Certifications
  - Training improves
-> Software Configuration Management
  - Version Control
  - Git, subversion, etc.
-> Analysis of Past-Projects
  - what went wrong
  - where most mistakes occur

PMI -> PMP
PMI -> ACP
Scrum Master

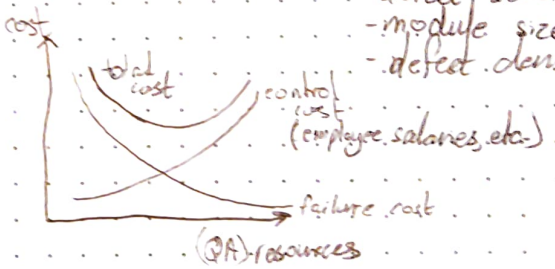
Management Components

-> Progress Control
  Managers do planning, monitoring, controlling
-> Quality Metrics
  - defect density
  - productivity
  - defect severity -> weighted defects
    -> module size

M1 | M2
100 500

Quality depends on:
- defect severity
- module size
- defect density

-> Cost
  - Minimize project cost
    - failure cost
    - control cost



Standards Components
-> Quality Management (What)
  - CMMI capability Maturity Model Integrated
    -> developed by CMU
  - ISO 9000:3
-> Project Process (How)
  - IEEE 1012
  - ISO/IEEE/IEC 12207

Human Components
→ Employees, team members, project managers.

CEO, CTO, CFO.
Top
Department Heads
Middle
Tiers
Bottom

Factors Affecting QA system

- organization size $\propto$ QA effort required
- project complexity $\propto$ QA effort required
- experience (relevant to the project domain) $\propto \frac{1}{\text{QA effort required}}$
- 3rd party engagement $\propto$ QA effort required

- acquaintance of team members with each other $\propto \frac{1}{\text{QA effort required}}$
- extent of reuse (with-of reuse) $\propto \frac{1}{\text{QA effort required}}$
- qualification $\propto \frac{1}{\text{QA effort required}}$

Question 03(b)

SQE LO7+08 11-09-2024
13-09-2024

BRING CALCULATOR USP (Unique Selling Point)

INTEGRATION OF SQA ACTIVITIES IN SDLC

Usability of drop-down vs. textbox
Recognition vs. Recall

Process Control vs. Quality Management: What vs. How, not product vs. process oriented

Software Processes

Process: a set of activities approached in a specific order.
↳ approach, roadmap,
ProcessModel: an abstraction of the process.

Spiral ∴ a meta-process

change-friendliness

Conventional, Rigorous, Heavy-weight

Light-weight, lean

Traditional

I Know It When I See It Syndrome

Processes

Agile

Evolutionary

Non-Evolutionary

User Acceptance Testing (UAT)

Recommends OOP

XP

Release

Planning

Pair programming

Customer Colocation

refactoring

KISS Minimalist Design

project velocity

Scrum

Gathe

CRC cards

User Stories

Committment
what user stories by when deadline?

no. of user stories implemented in the current release

Throw-away prototypes

Build-upon prototypes

IKIWISI

Prototyping

Planning Problem: when will it end

Spiral

Prototyping

Evaluation

Engineering

Planning

Risk-driven process

Heavy-weight

Big Bang Baking

Waterfall (Classical)

Incremental (Pure)

no. of features.

RAD

↳ Rapid Time, ~60-90 days

design prototypes called spike solutions

Large Projects

or medium

Small Projects

Stable Reqs.

Unstable Reqs.

Change welcome

only in hybrid versions

- large projects,
- stable reqs.
- inacciplable when:
- unsavy customer
- volatile reqs.
- feedback needed

Naturally Modularizable Problem

High Manpower

High Expertise

Rapid Delivery Required

- Kent Beck's book on XP.
- Refactoring by Martin Fowler

User Stories

As a who, I want what so that why

- Priorities (SC)
- Cost ($)
- value

February 2001, 17 people signed the agile manifesto in Utah.

↳ Kent Beck, Robert C. Martin

→ later extended to other fields like manufacturing, design, and project management.

Agile Manifesto

1-	Individuals & Interactions	vs.	Processes & tools
2-	Working software	vs.	Comprehensive Documentation
3-	Customer Communication	vs.	Contract Negotiation
4-	Responding to change	vs.	Following a plan

- **Scrum**: Managerial-heavy, as opposed to technical-heavy as in XP.

Iterations => Sprints
- → Time-boxed
  - → Scrum Poker (for estimation and planning).
- Requirements do not change during a sprint
- Backlog (outstanding undone requirements, prioritized)
  - → Product Backlog
  - → Sprint Backlog
- → In care of product-owner
- → Release (Demo)
- → Scrum Retrospective

Daily Scrum Meetings
- → Led by the scrum master
  - → Done standing up in the chairs
  - Three Questions:
    - What's the progress since the last meeting?
    - What will you do till the next scrum?
    - Are there any problems?

Traditionalists vs. Agilists
- Globe-eyed Hackers
- Stuck in the mud

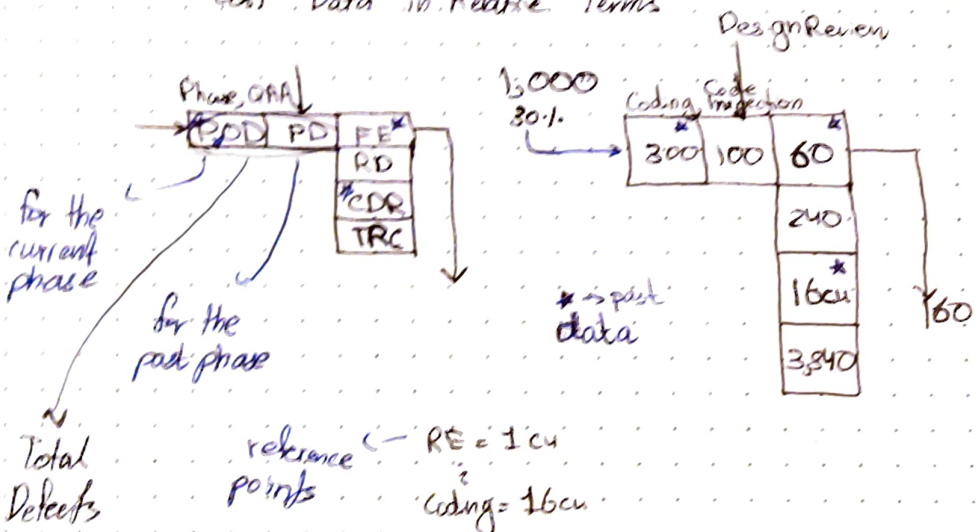
Barry Boehm → Balancing Agility and Discipline

SWE 203 (continued) 13-09-2024

Dimensions of SQA

Defect Removal Effectiveness & Cost Model

- QA activities are filters, not 100% accurate, but still cost something.
  - Defect
  - Filtering Effectiveness
  - Cost Data in Relative Terms



Verification: are we making the product right?

Validation: are we making the right product?

Qualification:
- coding conventions,
- good SE practices
- general standards

Circle Area Example:
- Verification: Is it programmed as πr^2?
- Validation: Did the customer actually need this?

$$RD = (POD + PD) \times \frac{FE}{100}$$

$$TRC = RD \times CDR$$

Past Data
- Phase Originated Defects (POD)
- Past Defects (PD)
- Filtering Effectiveness (FE)
- Removed Defects (RD)
- Cost of Defect Removal (CDR)
- Total Removal Cost (TRC)

Assumptions:
- All SQA activities are filters
- Past data is accurate and still relevant.

Question 03 (b)

SQE Corp

No.	Software Development Phase	originating % of defects in this phase
1	Requirements	35 %
2	Design	40 %
3	Coding (30% coding, 10% integration)	
4	Documentation	10%

Standard QA plan (Plan A) → Example: 1773.5 cu total

No.	QA activity	Filtering effectiveness	Cost of removing defects
1	Requirement specification Review	50%	1 cu
2	Design review	50%	2.5 cu
3	Unit Test - Code	50%	6.5 cu
4	Integration test	50%	16 cu
5	Documentation Review	50%	16 cu
6	System test	50%	40 cu
7	Operations phase	100%	110 cu

Comprehensive SQA Plan → Example: 1046.4 cu total

No.	SQA Activity	Filtering effectiveness	Cost of Removing defects
1	Requirement specification review	60%	1
2	Design inspection	70%	2.5
3	Design review	60%	2.5
4	Code inspection	70%	0.5
5	Code - ~~team~~ unit test	40%	6.5
6	Integration test	60%	16
7	Documentation review	60%	16
8	System test	60%	40
9	Operations Phase	100%	110

Contract Review

Pre-project component

Crime has Consequences

SS — SC

Legal contract
- handshake
- both parties are signatories

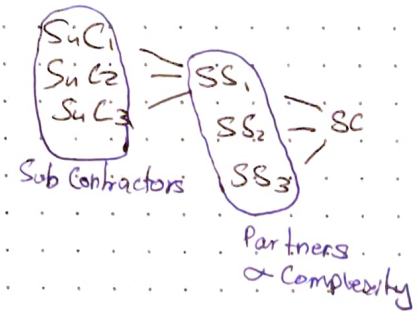
RFP → Proposal Draft → (submit) Proposal → Contract Draft → (if accepted) Contract
(Request for Proposal)
Proposal + Review
Contract Draft + Review

- Requirements clarified
  - ↳ scope
- Alternative Approaches
- Relationship Formality
  - Focal Person representative
  - Project deliverables
  - Change approval process
- Risks
  - technological
  - non-technical (personnel turnover, economic, political conditions)
- Estimates → estimation techniques
  - Effort
  - Duration
- Capacity / Capability
  - Can the supplier do it?
  - Can the consumer pay us?
- Participants
- Intellectual Property
  - Who gets to own the copyrights.

sign → a legal document → Contract

Formality benefits both sides by enforcing protocols.
↳ SQA activities should be scheduled.

SS ~~ SC
blurred when
- SS and SC belong to the same org.
- embedded systems
- products for customers



Goals of Contract Draft Reviews

- no unclarified issues.
- understandings documented
- no "new" changes

no ambiguity
no less
no more

extent of the contract draft review
- depends on project size, complexity, organisational complexity, acquaintance/familiarity/expertise

Characteristics of Bad Contracts

- unrealistic estimates
- loose requirements.

estimates can vary based on the software development process. But the choice of the process need not be documented.

SQE L11 25-09-2024

~~FORMAL TECHNICAL~~ REVIEWS → lifecycle components

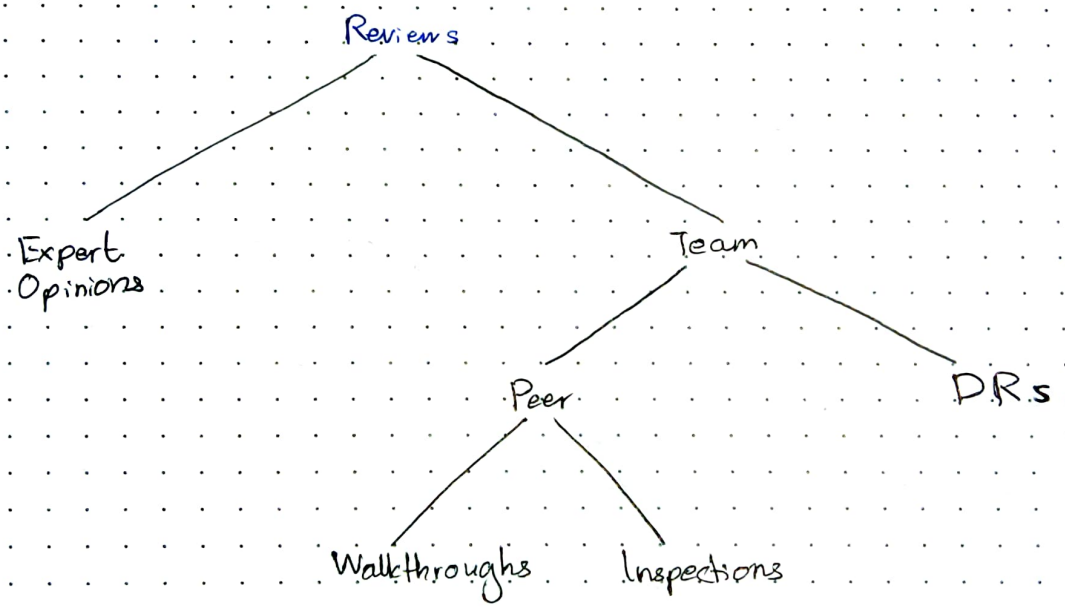
Review: critical evaluation/appraisal; formal
- Used in thesis reviews, FYP reviews
- double-blind review: no one knows who is the other person
- Reviews are used in other fields

- Reviews also focus on code, but primarily on docs.

- Reviews are generally good at detecting defects. They acts filters to remove defects.

Primary focus of:
- reviews
- Testing

Software =
- Programs
- Docs
- Data



Emphasis on objectives varies from review to review

Direct Objectives (deal with the project at hand ☝)

- detect and remove as many upstream faults (check incomplete, missing info/parts)
- identify new risks
  - older risks already identified in risk plan
- detect deviations from templates/conventions/standards
  - deviation cause problems downstream in maintenance
- grant or deny approval

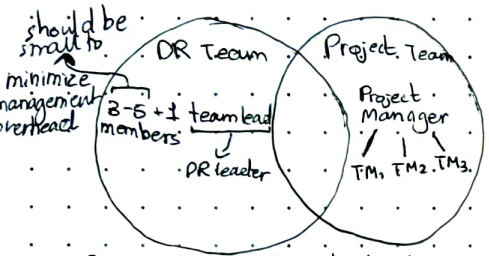
Indirect Objectives (deal with all projects)
- exchange professional knowledge (in meetings of professionals).
- record/analyze data related to faults

Expert Opinions
↳ Outside consultants, senior people

When outsiders are needed?
- when the SC is small and doesn't have enough manpower
- when the in-house expertise is not enough
- resolve disagreement

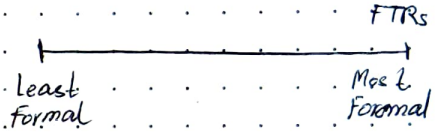
Design Reviews (a.k.a. Formal Design Reviews)
(a.k.a Formal Technical Reviews)
↳ most formal, most authoritative

4 segments of DR: ~~before~~ planning, before, during, after.



People may overlap, but play different roles.

The formality spectrum



Planning: assemble the team and appoint DR Leader.

DR leader:
- should not be the project manager of the project at hand
- should be preferably more senior than the project manager
- may be the project manager of another project
- should be familiar with the type of project

DR leader ensures:
- no personal criticisms
- schedule is followed
- offline discussion of internal conflicts
- external experts may be involved

DR team members:
- majority of these should not be from the project team
  - only senior people could join in a minority
- should be diverse (to get a diverse perspective of feedback)

Before
- DR leader assigns responsibilities and sets a schedule.
  - small projects may be reviewed in entirety by everyone, otherwise work will be divided.
- ~~DR is reviewed~~ Review documents against checklists
- DR team comes prepared with comments
- Project team prepares short presentation.

During
- DR session starts with a short presentation.
- The comments and action items are discussed along with assignees, due dates, and verification assignees.
- DR leader is usually the scribe
- Decision (full/partial approval, or denial) is ~~sent~~ made
- As many sessions may be held as needed, but one session should not last more than 2 hours.

After
- PR leader prepares a DR report quickly and shared with stakeholders. It is
- Follow-ups to reflect on the ~~process~~ DR.
- If something is not fixed satisfactorily another follow-up is needed.

PEER Reviews 27-09-2024

- Formality

Least → Most

Inspections — Walkthroughs — DRs

- Team composition:
  - Peers make-up a PR team
    ↳ peers of the author of the document

	Inspections	Walkthroughs → Quicker but less comprehensive.
Planning Phase	Leader: Moderator Team members: - Author(s) - a designer - a programmer - a tester - Project Manager should not be the leader	Leader: Coordinator Team Members: - Author(s) - a standards enforcer - a user representative - a maintenance person - Project Manager may be the leader
Before Phase	- Overview Meeting - Moderator should give an overview of the document being ~~rein~~ inspected if team members are outsiders. - Thorough Preparations: - Discuss Comments - Evaluate documents against checklists	- Brief Preparation - skimming the documents, - possibly giving comments
During Phase	- Short presentation - Programmer presents the document - Sessions like in DRs - No decisions taken	- Short presentation - document is presented by the authors - Sessions like in DRs - No decisions taken
After Phase	- 2 reports → findings → summary - Follow-up	- 1 report → findings - no follow-up

Effort	vs.	Duration
- person hours - PDs - PW - PM - PY		- hours - days - weeks - months - years

SQE 02-10-2024

Software Testing Strategies

Inc RE A D I T D/D C-M O/R

Testing

- the classical, oldest, last-line-of-defense against faults.
- takes the most amount of SQA resources.
- Testing requires execution of code
  - not every execution of code ~~requires~~ is testing
- Testing requires a certain mindset that faults need to be uncovered.

Coders:
- programmable or moonshot?

Testers:
- quantified
- reqs. testable?

Designers:
- use-cases fishy w.r.t GUI
- modularity, coupling, cohesion
- functional independence

Strategy vs. Technique

Software	Testing	Strategies	Techniques
-programs/code -data -documentation	-execution -intention to find faults -formal -approval -third party (optional)	-what -who -when	-how

→ high-level
→ low-level

Objectives of Testing

Direct:
- identify as many faults as possible and fix them
- do this efficiently
- acceptable quality

Indirect:
- to compile a record of fault data

Software Testing Strategies

General Testing Strategy

Unit Testing
- ~~test integ. related units~~
- test a single unit
- unit = set of related classes/functions
- data structures - UI
- algorithms
- loops
- conditions

Integration Testing
- test integrated modules.
- big-bang approach
- incremental approach
  - ~~TD~~ Top-down - Depth first
  - Bottom-up - Breadth first

Validation Testing
- test conformance to requirements

System Testing
- non-functional requirements
- performance test intended
- stress test on ~~related~~ hardware
- more related to embedded systems.

SQE 04-10-2024

The Hawthorne Effect

Decision-Makers vs. Work-forces

Integration Testing

Big-Bang vs. Incremental

Big-Bang:
- Integrate everything together at the same time.
- Works well for small projects.
- Pinpointing the errors in the modules is more difficult.

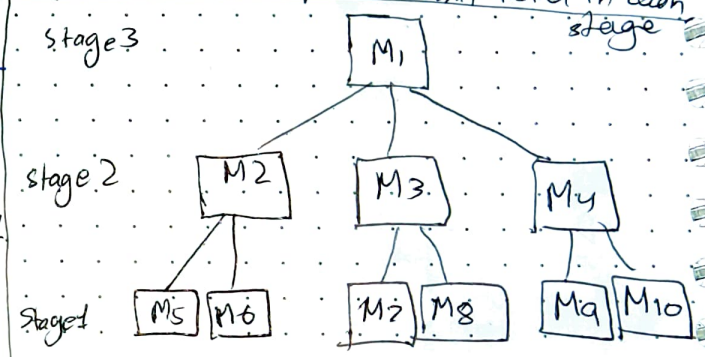
Incremental:
- Requires "Dummy" Modules
- Integrate step-by-step.
- Overhead Cost.
- Which one to use? Follow the development approach.

Top-Down ("Stubs") vs. Bottom-Up ("Drivers")

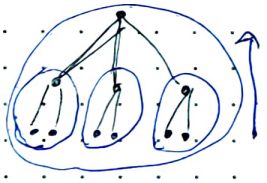
- "Details" "Error Message" are ambiguous
- Database, backend things should not be present in the SRS.
- Formatting issue should be mentioned.
-

User Acceptance Testing (UAT)
- Bespoke/custom products require UAT.
- Done in natural environment

Develop and unit tested in each stage



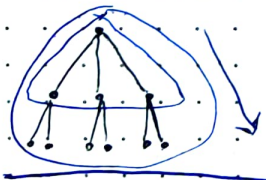
Bottom-Up Breadth-First



Bottom-Up Depth-First

Depth-First: single features are tested earlier in their entirety.

Top-Down Breadth-First



Top-Down Depth-First

Smoke Testing:
- Electronics ~~short-circuit~~ ~~delay~~ release smoke only when there's a short-circuit.
- tests whether
  - software runs
  - does not crash
  - major features work (no show-stoppers)

aka
- "Build-Verification Testing"

Validation Testing:
- if end users involved; not professional testers.
  - alpha testing
    - end-users test the software in a controlled environment.
    - differentiation from Hawthorne Effect
  - beta testing
    - end-users test the software in their natural environment.
    - used for generic/open-market products.

Regression Testing:
- Done after each change
- Test subset of previous test cases to check if the change hampers previously working code.

SQE 09-10-2024.

Software Tests

NCCS

consume the most SQA resources.

consumes the most testing resources

Concept

The Box is the "software".

- BlackBox (BB) a.k.a functional testing
- WhiteBox (WB) Glass-Box, a.k.a structural testing — code/docs + data
- Hybrid Versions, like Grey-Box testing.

Quality Factors

- Operation
  - Correctness
  - Reliability
  - Efficiency
  - Integrity
  - Usability
- Revision
  - Maintainability
  - Flexibility
  - Testability
- Transition
  - Interoperability
  - Portability
  - Reusability

- Correctness: WB/BB
  - correctness of docs (like manuals, user, installation manuals, etc.)
  - correctness of outputs
    - ~~for~~ with/without focusing on details
  - availability testing
    - usually done with load testing
    - response time as a function of load
    - usually automated

- Reliability (BB)
  - MTBF between failures
  - MTTR to repair
  - MDTM down-time per month
  - Statistical models exist for reliability testing

- Efficiency: (BB)
  - Stress tests
    - load testing + ~~duration~~
      - behavior/performance of the system under abnormal load.
    - durability testing
      - expose system to abnormal physical conditions.

- Integrity/security testing (mostly BB)
  - Malware, viruses, trojans
  - SQL injection
  - ethical hacking

- Usability Testing (mostly BB) - difficulties increase error chances
  - Training usability testing
  - Operational usability testing

- Maintainability (BB/WB)
  - corrective
  - modular design, functional
- Flexibility (BB/WB) independence, low coupling, high-cohesion.
- Testability (BB)
  - intermediate results
  - ~~bits~~ in-house checks
- Portability (BB)
  - The software works on other systems or not
- Reusability (WB)
  - Satisfies criteria for inclusion in the re-use library
  - Coupling, cohesion, functional independence
- Interoperability (BB)
  - Working with other softwares/interfaces/systems.

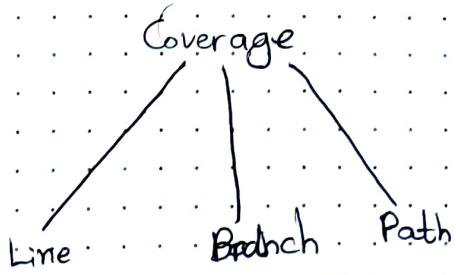
SQE 11-10-2024

WHITE BOX TESTING

If there are 10 sequential, un-nested conditionals, there are $2^{10} = 1024$ possibilities.

EXHAUSTIVE TESTING IS NOT PRACTICAL

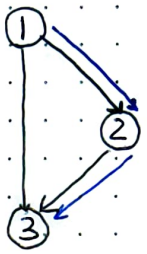
- selective testing is preferred, unless the module is very critical.



- All programming language statements must be executed at least once.
  - ↳ Line coverage = 100%

100% line coverage ---does not guarantee---> 100% branch coverage

100% branch coverage ---guarantees---> 100% line coverage



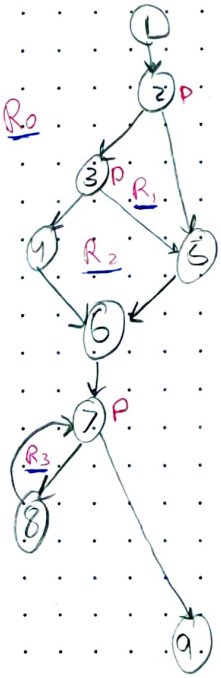
This test case covers 100% of the lines, without covering all branch.

Each branch is

Path > Branch > Line

Basis Path Testing / Basic Path Testing
↳ Guarantees 100% Branch Coverage.

Code → Program Flow Graph (G) (--> has certain rules) → Cyclomatic Complexity V(G) (--> Upper bound on the no. of basis paths.) → Basis Paths → Voila! 100% Branch coverage.



Regions — Enclosed Region

V(G) — R = ER + 1 = 3 + 1 = 4

V(G) — P + 1 = 3 + 1 = 4

V(G) — E - N + 2 = 11 - 9 + 2 = 4

Edges / Nodes

Predicate Nodes: Nodes with 2 or more outgoing edges

Upper-bound on the amount of basis paths needed for 100% Branch Coverage.

It is a UPPER BOUND it is possible to achieve 100% branch coverage with lesser test-cases.

lefty-truthy, righty-falsey.

Rules

1. Bunch together all consecutive sequential non-branching statements in a single node.
2. All loop/if/else conditions become one node.

SQE L17 16-10-2024

IDEs may calculate the cyclomatic complexity.
- It can be done by identifying the nodes and creating an nxn adjacency matrix for the control flow graph.

V(G) is used for test prioritization

Set of Basis Paths

S#	Path
1.	1→2→5→6→7→9
2.	1→2→5→6→7→8→7→9
3.	1→2→3→4→6→7→9
4.	1→2→3→5→6→7→9

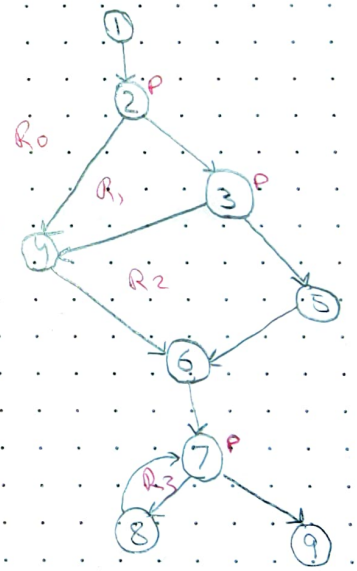
V(G) is an upper limit: there may be less than V(G) paths which guarantee 100% branch coverage

→ Independent (by definition)
→ Subsequent paths remain independent by adding at least one such edge which is not present in any of the previous/former independent paths.

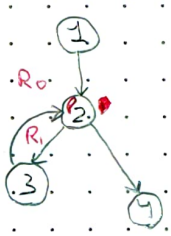
S#	Path
1.	1→2→3→4→6→7→8→7→9
2.	1→2→5→6→7→9
3.	1→2→3→5→6→7→9

V(G) is the cyclomatic complexity, which is to be used for calculating the Weighted Methods per Class (WMC) in the CK set of OO-Metrics.
$WMC = \sum c_i$

S#	Path
1.	1→2→3→5→6→7→8→7→9
2.	1→2→3→4→6→7→8→7→9
3.	1→2→3→5→6→7→9

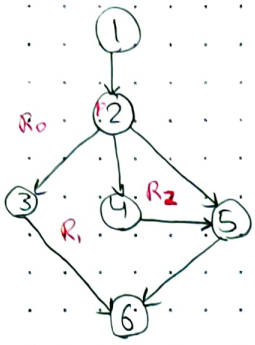


S#	Path
1.	1→2→3→2→4




```
void foo(){
  //...
  for(int i=0; i<10; i++){
    //...
  }
  //...
}
```

V(G)
= R = ER+1 = 1+1 = 2
= E − N+2 = 4−4+2 = 2
= P+1 = 1+1 = 2



$V(G) = R = ER + 1 = 2 + 1 = 3$
$= P + 1 = 1 + 1 = 2$
$= E - N + 2 = 7 - 6 + 2 = 3$
$\Rightarrow V(G) = 3$

```
int selection (int option){
  int value;
  int denominator;
  switch(option){
    case 1:
    value = option++;
    denominator = 2;
    break;

    case 2:
    value = option--;
    denominator = 4;
    default:
    value = 0
    denominator = 1;
    break;
  }
  return (value/denominator);
}
```

(1)

```
void foo()
{
  ...
  bar();
  ...
}
```

function calls are non-branching.

```
void bar(){
  ...
}
```

- Recursive calls are just like function calls.

SQE L18 18-10-2024

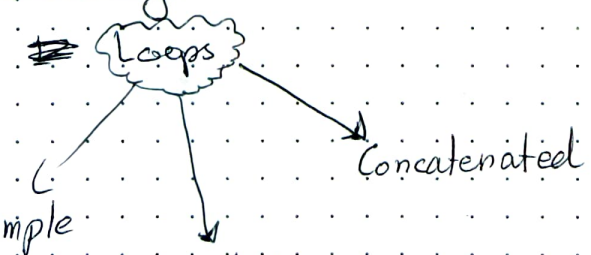
Testing

Condition Statements

↳ have more error chances
↳ if-else-if/switch
↳ use operators
- relational >, <, >=, <=, ==
- ~~logical~~ logical &&, ||, !
  - incorrect operators
  - missing operators
  - parantheses

Boris Beizer: "Bugs lurk in corners and congregrate at boundaries"

Loop Testing:



↳ skip the loop
↳ have min iterations (n-levels)
↳ have max iterations
↳ have somewhere in between the range iterations
↳ more than max. iterations

Nested loops:

a- Begin with inner loop. keep all outer loops at min. iterations
b- Move out one loop at a time.

Concatenated:

Independent scenario:

Do simple testing of each loop in sequence.

Dependent Scenario:

Treat loops as nested.

Advantages of White Box Testing

- Test source code directly
- Coverage can be controlled directly
- Qualification Testing (conformance to standards) can also be tested.

Disadvantages:

- Resource-heavy, takes more time than black-box testing.
- Options are limited

SQE
23-10-2024

~~Chapter 6:~~

~~Floyd's Algorithm~~

Black-Box Correctness Testing

↳ since source code is not available, we test the correctness of the outputs.

Equivalence Class Partitioning (ECP)

Input is partitioned into equivalence classes.

Equivalence Class: a set which contains inputs which make the software behave in the same manner.

- created for each variable, keeping in mind the software.

CGPA
B1) 0.0 - 1.9 → 0.5, 1.2, etc.
B2) 2.0 - 4.0 → 2.1, 3.7, etc.

Invalid Equivalence Classes: sets of input values to which the software responds with specific error messages. Each type of invalid ~~e~~ input will have a separate equivalence class.

Coverage in Correctness Testing:

We need minimum test-cases

Rules:
1- Write a separate test case for each invalid e.class and test only on e.c in each test case.
2- Write a single test case for all the valid e.classes together.

The idea is that separately checking incorrect inputs easily identifies the underlying error.

→ Black-box test cases may be written even before the code is available. Although code is needed to execute test-cases, it is not needed to 'write' these test-cases.

Boundary Value Analysis (BVA)

The boundary values of all equivalence classes used for testing.

- can not be used for discrete variables, like gender, types, etc.

SC Example
↳ Swimming Centre

- Day of the week → Weekday / Week-end day
- Visitor's status → Member / One-time visitor
- Visit Time → Shift-1 / Shift-2
- Age → Young / Middle / Elderly

Possible Valid Combinations = 2×2×2×3 = 24

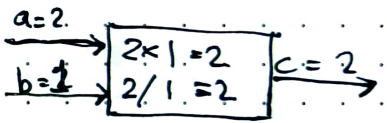
- Costs given in table 9.7, Galin.

Number of test-cases in 1A = maximum no. of ~~input~~ equivalence classes of any input variable

" in 1B = maximum no. of boundaries

" in 2 = no. of invalid equivalence classes.

= twice the number of equivalence classes of a continuous input variable with the maximum no. of equivalence classes



Advantages

- Cheaper than whitebox testing (where both can be used)
- Many options

Disadvantages

- Qualification testing not possible
- No control on coverage
- We may get the correct output by chance

no fine control on code coverage. Some control on conceptual coverage.

SQE 30-10-2024

THE TESTING PROCESS

Decide Methodology → Plan → Test Case Design (1-5) A Damage Level (1-5) B Risk Level → Execute (Reports are prepared perhaps automated.)

- Acceptable level of quality
- Strategies & Techniques

- What do we test?
- Who performs the tests? ss/sc/consultants
- Where is the testing performed? ss/sc/consultants
- When do we stop testing? How much testing is enough?

High-level decisions to be made at start of the project:

- Acceptable Level of Quality
- Strategies & Techniques
  - Incremental / Big-Bang
  - Top-down / bottom-up
  - Breadth-first / Depth-first

Stopping:

- stop when all is tested
  - assumes unlimited budget
  - may work for tiny projects.
- stop when we have run out of money/time
  - surprisingly, very common.
- stop based on mathematical models
  - a threshold is defined for the "defect detection rate" based on past data
  - criticisms:
    - different severity-level defects treated equally
    - current data not reliable predictor of future data
- fault seeding
  - intentionally inject defects
  - define threshold
  - premise: correspondence between seeded and actual defects.
  - seeded faults removed before delivery.

A → How dangerous is the damage?
B → How likely is the damage.

i- A+B
ii- kA + mB
iii - A * B ← Risk Exposure (Expected damage)

Used in project management to prioritize project risks.

- dual independent testing teams.

TEST CASE DESIGN

Test Case: a set of input values along with operating conditions and an expected output.

At least 1 test case for each course of action of every use-case.

Priority = {"High", "Medium", "Low"},

Dual Independent Testing Teams Method

Na → no. of faults detected by team A
Nb → no. of faults detected by team B
Nab → no. of faults detected by both teams
N → total no. of faults
T → threshold

$$P_a = \frac{N_a}{N} = \frac{N_a}{\frac{N_a \times N_b}{N_{ab}}} = \frac{N_{ab}}{N_b}$$

$$P_b = \frac{N_b}{N} = \frac{N_b}{\frac{N_a \times N_b}{N_{ab}}} = \frac{N_{ab}}{N_a}$$

$$P_{ab} = \frac{N_{ab}}{N} = P_a \times P_b = \frac{N_a}{N} \times \frac{N_b}{N} = \frac{N_{ab}}{N}$$

$$\frac{N_a \times N_b}{N_{ab}} = \frac{N^2}{N} = N$$

$$N_{(a)(b)} = P_{(a)(b)} \times N$$

$$P_{(a)(b)} = (1 - P_a)(1 - P_b)$$

$$= \left(1 - \frac{N_a}{N}\right)\left(1 - \frac{N_b}{N}\right)$$

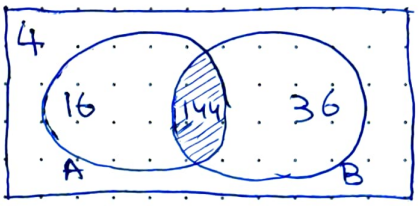
$$= \frac{(N - N_a)(N - N_b)}{N^2}$$

$$= \left(1 - \frac{N_{ab}}{N_b}\right)\left(1 - \frac{N_{ab}}{N_a}\right)$$

$$= \frac{(N_b - N_{ab})(N_a - N_{ab})}{N_a \times N_b}$$

$$= \frac{\left(\frac{N_a \times N_b}{N_{ab}} - N_a\right)\left(\frac{N_a \times N_b}{N_{ab}} - N_b\right)}{\left(\frac{N_a \times N_b}{N_{ab}}\right)^2}$$

$$= \frac{(N_a \times N_b - N_a \times N_{ab})(N_a \times N_b - N_b \times N_{ab})}{(N_a \times N_b)^2}$$



Example:

Na = 160
Nb = 180
Nab = 144
T = 2.5% = 0.025

$$P_{(a)(b)} = \left(1 - \frac{160}{200}\right)\left(1 - \frac{180}{200}\right)$$

$$= \left(\frac{40}{200}\right)\left(\frac{20}{200}\right)$$

$$= 0.02 = 2\%$$

since 2% < 2.5%, testing may stop

$$N_{(a)(b)} = P_{(a)(b)} \times N$$

$$= 0.02 \times 200$$

$$= 4$$

$$\frac{N_{ab}}{N} = \frac{144}{200} = 0.72$$

$$P_a \times P_b = \frac{160}{200} \times \frac{180}{200} = 0.72$$

Using simplified formula:

$$P_{(a)(b)} = \frac{(180 - 144)(160 - 144)}{160 \times 180}$$

$$= 0.02$$