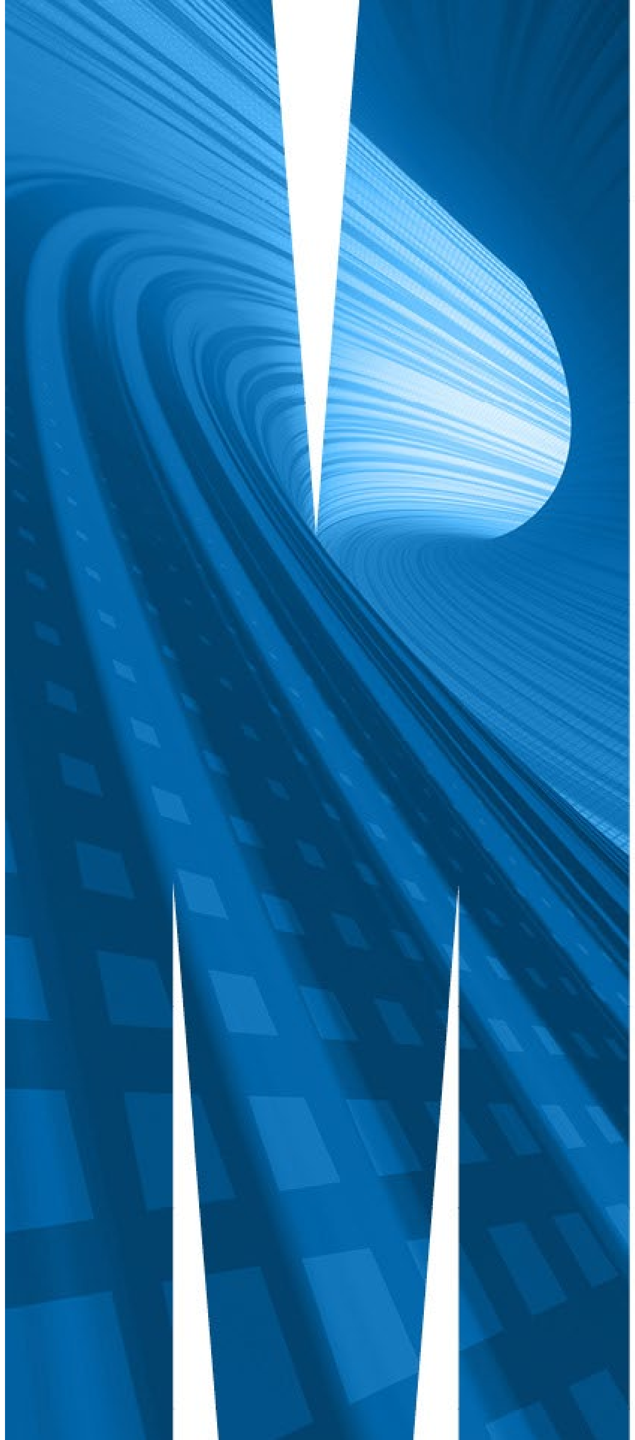
MONASH University

FIT2107-Software Quality & Testing

Lecture 12 – Exam Information & Revision

3rd November 2020

Dr Najam Nazar



Exam

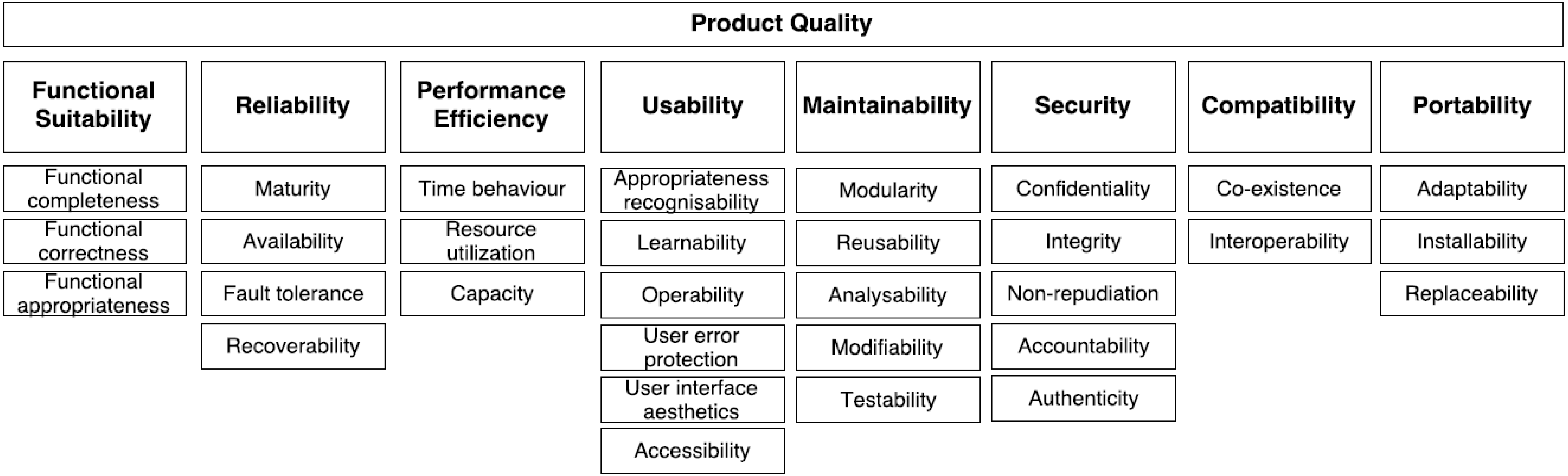
- You already know
  - Closed Book/Invigilated.
  - 2 hours 10 minutes for reading and attempting the exam
  - 30 minutes extra to upload answers for Hybrid Questions.
- Everything covered in Lectures that includes Videos and slides, Workshops and Reading Resource (Mandatory) is examinable.
- Mock exam already shared so do attempt it.
- Past Exams will be uploaded as well for practice. However, they may have questions that are not part of syllabus for this semester, so you don’t need to practice those questions.
- https://www.monash.edu/exams/electronic-exams/rules

REVISION

Week 1: Software Quality

- Quality means the degree to which a product or a process meets requirements (functional quality).
- Supports the delivery of functionality requirements such as maintenance, robustness etc. (non-functional quality).
- What makes software quality so hard?
  - Software systems are growing in **complexity**: Modern systems are composed of millions of lines of code!
  - Some **quality requirements** are difficult to specify in an unambiguous way
  - Software systems are **intangible**: Our senses cannot help us understand them
  - Software systems are **malleable**: Small changes can have huge repercussions
  - **Limited human resources** for finding defects: Window for finding/fixing defects is small

Week 1: Software Quality Attributes

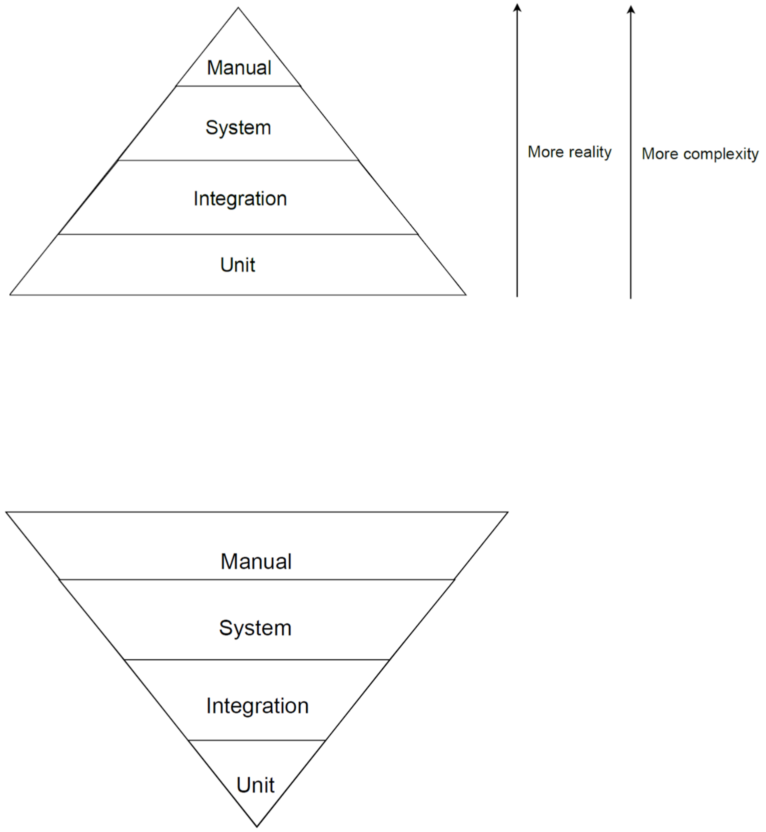


Product Quality

Functional Suitability	Reliability	Performance Efficiency	Usability	Maintainability	Security	Compatibility	Portability
Functional completeness	Maturity	Time behaviour	Appropriateness recognisability	Modularity	Confidentiality	Co-existence	Adaptability
Functional correctness	Availability	Resource utilization	Learnability	Reusability	Integrity	Interoperability	Installability
Functional appropriateness	Fault tolerance	Capacity	Operability	Analysability	Non-repudiation		Replaceability
	Recoverability		User error protection	Modifiability	Accountability		
			User interface aesthetics	Testability	Authenticity		
			Accessibility				

Week 2 – Testing

- Testing Perspective
- Objectives
  - Functional Correctness
  - User Acceptances
  - Performance
  - Security
  - Reliability
  - Robustness
  - Regression
- Testing Pyramid and Cones



Week 2 – Testing

- Failure: A component or a system behaves in a way that is not expected.
- Defect/Bug/Fault: flaw in a component that can cause a system to behave incorrectly
- Error/Mistake: action that produces an incorrect result.

- A mistake by a developer leads to a fault in the source code that will eventually result in a failure.

Week 3 & 4: Blackbox Testing

Input
Black Box
Output

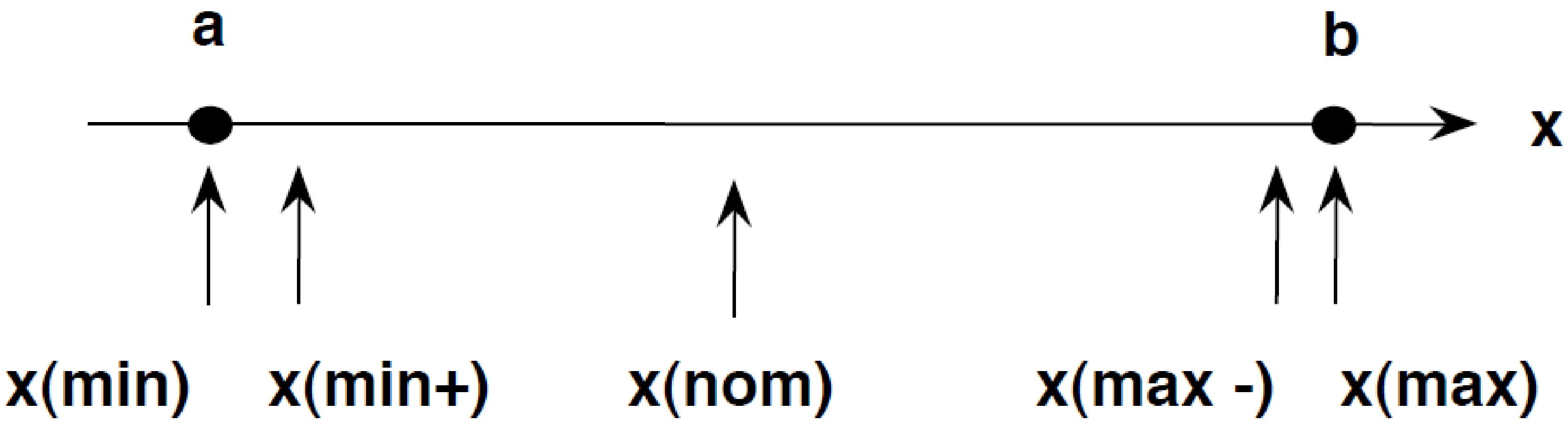
- Testing by generating random inputs is random testing.
- Partitions: A program may behave differently under different conditions. So we need to test all behaviours.
    - Disjoint classes: no two partitions represent the same behavior.
    - Can easily verify if the behavior is correct or not.

Week 3 & 4: Blackbox Testing

- Equivalence partitioning
  - If the program behaves correctly for one given input, it will work correctly for all other inputs from that partition.
  - Valid and invalid partitions
- It is a systematic way of identifying test cases using the characteristics of the input values.
  - Categories – attributes that you'd divide up the input space
  - Choices – the groups of values that a category might take.

Week 3 & 4: Blackbox Testing

- Boundary Value
  - Testing between extreme ends or boundaries between partitions of the input values



Combinatory Testing

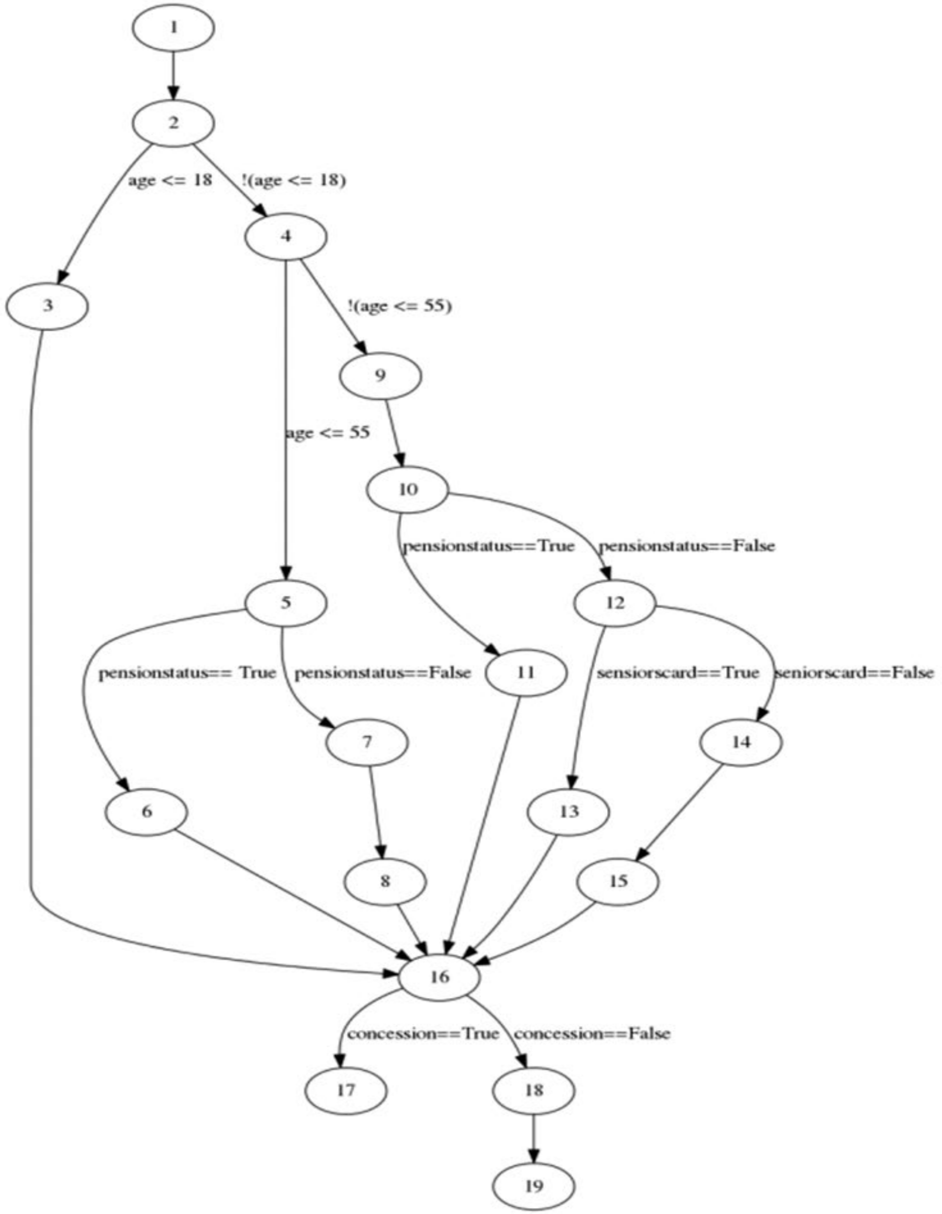
Pairwise Testing

- In pairwise testing, we design test cases to execute all possible discrete combinations of each pair of input parameters

Tests	OS	Browser	Protocol	CPU	DBMS
1	XP	IE	IPV4	Intel	MySQL
2	XP	Firefox	IPV6	AMD	Sybase
3	XP	IE	IPV6	Intel	Oracle
4	OS X	Firefox	IPV4	AMD	MySQL
5	OS X	IE	IPV4	Intel	Sybase
6	OS X	Firefox	IPV4	Intel	Oracle
7	RHL	IE	IPV6	AMD	MySQL
8	RHL	Firefox	IPV4	Intel	Sybase
9	RHL	Firefox	IPV4	AMD	Oracle
10	OS X	Firefox	IPV6	AMD	Oracle

Week 5 & 6: Whitebox Testing

- Tests internal structures or workings of an application, as opposed to its functionality
- Coverage is the amount (or percentage) of code that is exercised by the tests.
- For a given set of input data, the program unit executes a certain path.
- CFG is a graph representation of all paths that might be traversed through a program during its execution (nodes and directed edges).



Week 5 & 6: Whitebox Testing

Line/Statement Coverage

- requires every possible statement in the code to be tested

$$linecoverage = \frac{\#\ of\ lines\ covered}{Total\ \#\ of\ lines} * 100$$

Week 5 & 6: Whitebox Testing

Branch/Decision Coverage

- Branch Coverage requires every possible branch (i.e., if-else, other conditional loops) in the code to be tested.
  - two possible outcomes, true and false.
  - Both outcomes must be covered

$$branchcoverage = \frac{\#\ of\ executed\ branches}{Total\ \#\ of\ branches} * 100$$

Week 5 & 6: Whitebox Testing

Branch/Decision Coverage

- Branch Coverage requires every possible branch (i.e., if-else, other conditional loops) in the code to be tested.
  - two possible outcomes, true and false.
  - Both outcomes must be covered

$$branchcoverage = \frac{\#\ of\ executed\ branches}{Total\ \#\ of\ branches} * 100$$

Week 5 & 6: Whitebox Testing

(Basic) Condition Coverage

- When branches become more complicated it contains more decisions
- So branch coverage is not enough to test all possible cases.
- Conditions are tested separately and not the “big decision block”.
- Conditions and Branches are tested together.

$$conditioncoverage = \frac{conditions\ outcome\ covered}{Total\ \#\ of\ codition\ outcomes} * 100$$

Week 5 & 6: Whitebox Testing

Path Coverage

- All possible control paths taken, including all loop paths taken zero, once, and multiple.
- All paths in CFG

$$pathcoverage = \frac{paths\ covered}{Total\ paths} * 100$$

Week 5 & 6: Whitebox Testing

MC/DC

- Exercise each condition in a way that it can, independently of the other conditions, affect the outcome of the entire decision.
- Every possible condition of each parameter must have influenced the outcome at least once.

Week 7: Unit Testing

- A unit test is a way to verify expected functionality in a small, independent bit of code.
- A test might focus on one method (function) in a class.
- One method one test.
- Tests are divided into suites.
- Coverage.py is a third-party tool for measuring code coverages in Python programmes.
- It provides very nice command line and HTML output along with advanced features such as branch coverage.

Week 8: GIT & CI

- Software is built by teams of people working on the system simultaneously.
- Simultaneous changes, updates, enhancements and Tests etc.
- Version Control systems
- Gitlab
  - Clone, push, pull, commit…
- Continuous Integration (CI) is a development practice where developers integrate code into a shared repository frequently, preferably several times a day.
- Each integration can then be verified by an automated build and automated tests.
- While automated testing is not strictly part of CI it is typically implied.
- Continuous Integration doesn’t get rid of bugs, but it does make them dramatically easier to find and remove.

Week 9: Mocking

- mocks simulate the behavior of the real objects.
- Database calls
- Webservices
- unittest.mock

Week 10: Code Reviews

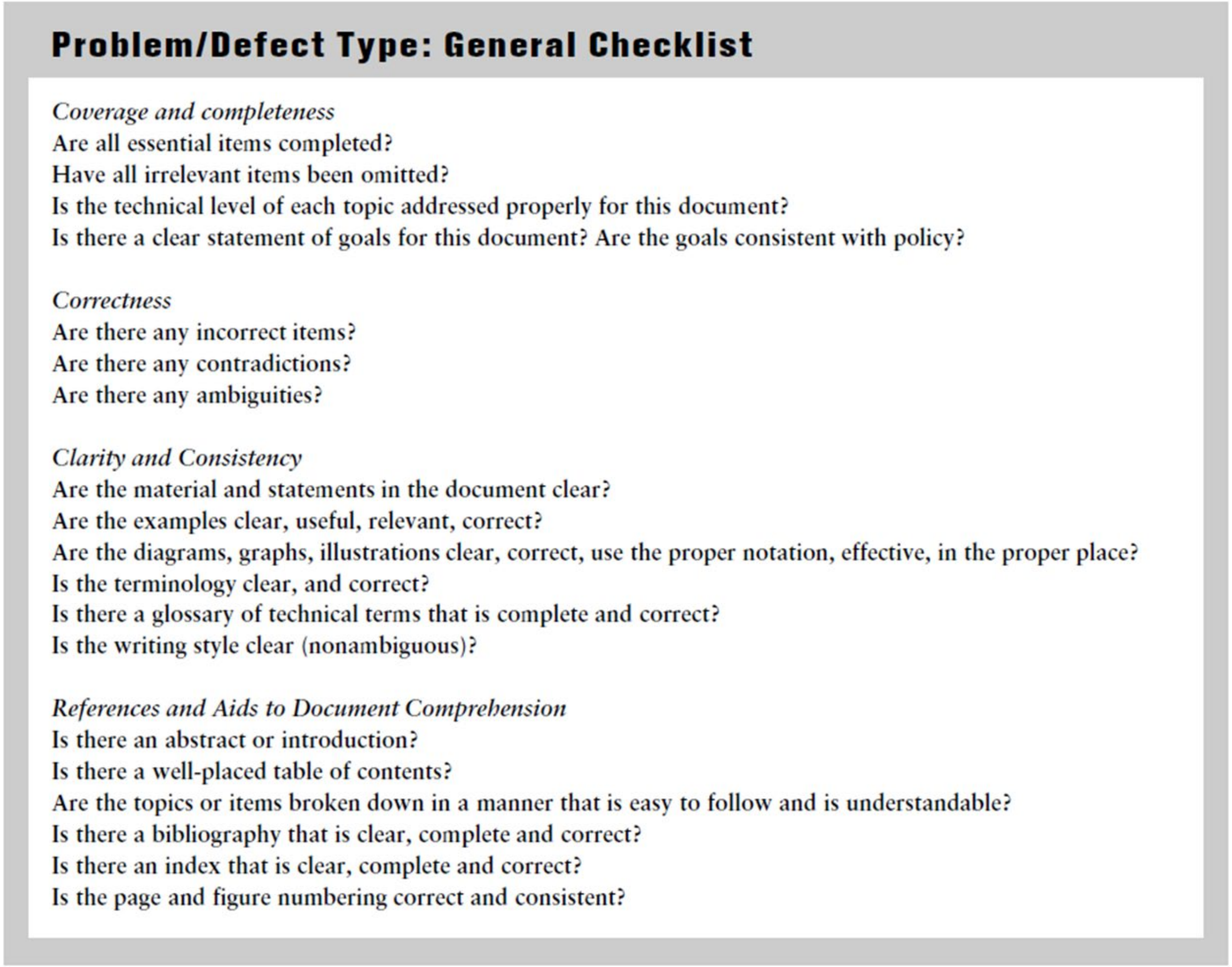
- Reviews: activities to evaluate software artefact through manual scrutiny
  - Inspections: Formally defined
  - Walkthroughs: Author defined.
- Code Reviews
  - Software quality assurance activity in which one or several people check a program mainly by viewing and reading parts of its source code.
- Main objective
  - Detect faults

Week 10: Code Reviews

- Moderator: guides the pace of the review process, selects the reviewers and schedules the review meetings, Objectivity.
- Author(s): has written the code to be reviewed.
- Presenter: other than the author of the code.
- Record Keeper: documents the problems found during the review process, automated.
- Reviewer(s): experts in the subject area of the code under review.
- Observer: Observe

Week 10: Code Reviews

Checklists



Problem/Defect Type: General Checklist

Coverage and completeness
Are all essential items completed?
Have all irrelevant items been omitted?
Is the technical level of each topic addressed properly for this document?
Is there a clear statement of goals for this document? Are the goals consistent with policy?

Correctness
Are there any incorrect items?
Are there any contradictions?
Are there any ambiguities?

Clarity and Consistency
Are the material and statements in the document clear?
Are the examples clear, useful, relevant, correct?
Are the diagrams, graphs, illustrations clear, correct, use the proper notation, effective, in the proper place?
Is the terminology clear, and correct?
Is there a glossary of technical terms that is complete and correct?
Is the writing style clear (nonambiguous)?

References and Aids to Document Comprehension
Is there an abstract or introduction?
Is there a well-placed table of contents?
Are the topics or items broken down in a manner that is easy to follow and is understandable?
Is there a bibliography that is clear, complete and correct?
Is there an index that is clear, complete and correct?
Is the page and figure numbering correct and consistent?

Week 10: Code Reviews

Modern Code Reviews

- Inspection/reviews
- Lightweight
- Few Formal Requirements
- Shorter in time
- Iterative.

Week 10: Code Reviews

Authors Guidelines

- Be Humble
- Your code doesn’t represent you as a person
- The reviewer and you are friends
- IKEA Effect (Cognitive Bias)
- Be Open to others’ perspectives.
- Bandwagon Effect

Week 10: Code Reviews

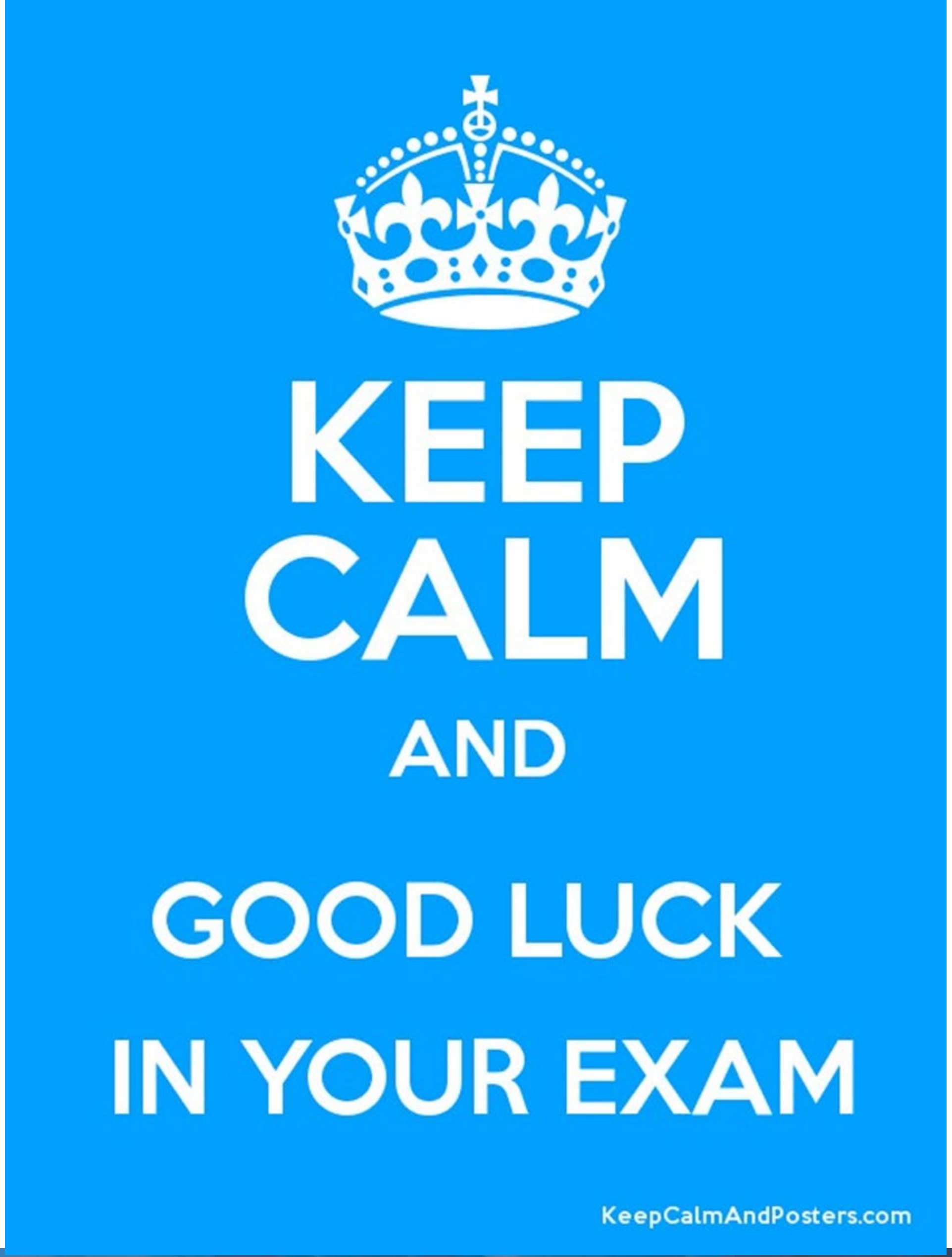
Reviewers Guidelines

- Focus on reviewing the code only
- Use I-Messages
- Prefer asking questions
- OIR Rule of giving Feedbacks
- Accept that there are different solutions
- Don’t jump in front of every train
- Praise

Week 11 – Software Metrics

- Important to get metrics
- Evidence based decision making
- LOC
- Coverage
- McCabe Complexity
- Halstead
- OOP Metrics
- Fault Prediction
- Taylorism

That's all Folks!


KEEP
CALM
AND
GOOD LUCK
IN YOUR EXAM
KeepCalmAndPosters.com