Static and dynamic Testing

Static testing

- Verifying the conformance of a software system and its specification without executing the code (by a computer)
- static testing = human testing

Static testing

- Involves analyses of source text by humans
- Can be carried out on ANY documents produced as part of the software process
- Discovers errors early in the software process
- Usually more cost-effective than testing for defect detection at the unit and module level
- Allows defect detection to be combined with other quality checks

Static testing effectiveness

- More than 60% of program errors can be detected by informal program inspections (Meyers: 30 - 70%)
- More than 90% of program errors may be detectable using more rigorous mathematical program verification
- The error detection process is not confused by the existence of previous errors

Program inspections

- Formalised approach to document reviews
- Intended explicitly for defect DETECTION (not correction)
- Defects may be logical errors, anomalies in the code that might indicate an erroneous condition (e.g. an uninitialised variable) or non-compliance with standards
- group code reading → team-based quality

Inspection pre-conditions

- A precise specification must be available
- Team members must be familiar with the organisation standards
- Syntactically correct code must be available
- An error checklist should be prepared
- Management must accept that inspection will increase costs early in the software process
- Management must not use inspections for staff appraisal
The inspection process

- Planning
- Overview
- Individual inspection
- Inspection meeting
- Rework
- Follow-up

Inspection procedure

- System overview presented to inspection team
- Code and associated documents are distributed to inspection team in advance
- Inspection takes place and discovered errors are noted, no repair
- Modifications are made to repair discovered errors
- Extension of checklists
- Re-inspection may or may not be required

Inspection teams

- Made up of at least 4 members
- Author of the code being inspected
- Reader who reads the code to the team
- Inspectors who finds errors, omissions and inconsistencies
- Moderator who chairs the meeting and notes discovered errors
- Other roles are Scribe and Chief moderator
- No superior

Inspection rate

- 500 statements/hour during overview
- 125 source statement/hour during individual preparation
- 90-125 statements/hour can be inspected
  - Meyers: 150 statements/hour
  - Balzert: 1 page/hour
- Inspection is therefore an expensive process
- Inspecting 500 lines costs about 40 man-hours effort = £2800

Inspection checklists

- Checklist of common errors should be used to drive the inspection
- Error checklist may be programming language dependent
- Complement static semantics checks by compiler + static analyser
- The 'weaker' the type checking, the larger the checklist
- Coding standard / programming guidelines

Inspection checks

- Data faults
  - Are all program variables initialised before their values are used?
  - Have all constants been named?
  - Should the lower bound of arrays be 0, 1, or something else?
  - Should the upper bound of arrays be equal to the size of the array or size - 1?
  - If character strings are used, is a delimiter explicitly assigned?

- Control faults
  - For each conditional statement, is the condition correct?
  - Is each loop certain to terminate?
  - Are compound statements correctly bracketed?
  - In case statements, are all possible cases accounted for?

- Input/output faults
  - Are all input variables used?
  - Are all output variables assigned a value before they are output?

- Interface faults
  - Do all function and procedure calls have the correct number of parameters?
  - Do formal and actual parameter types match?
  - Are the parameters in the right order?
  - If components access shared memory, do they have the same model of the shared memory structure?

- Storage management faults
  - If a linked structure is modified, have all links been correctly reassigned?
  - If dynamic storage is used, has space been allocated correctly?
  - Is space explicitly deallocated after it is no longer required?

- Exception management faults
  - Have all possible error conditions been taken into account?