

Code Reviews & Inspections

CSC 4700 Software Engineering

Software inspections

- These involve people examining the source representation with the aim of discovering anomalies and defects.
- Inspections not require execution of a system so may be used before implementation.
- They may be applied to any representation of the system (requirements, design, configuration data, test data, etc.).
- They have been shown to be an effective technique for discovering program errors.

Inspection success

- Many different defects may be discovered in a single inspection. In testing, one defect may mask another so several executions are required.
- They reuse domain and programming knowledge so reviewers are likely to have seen the types of error that commonly arise.

Inspections and testing

- Inspections and testing are complementary and not opposing verification techniques.
- Both should be used during the V & V process.
- Inspections can check conformance with a specification but not conformance with the customer's real requirements.
- Inspections cannot check non-functional characteristics such as performance, usability, etc.

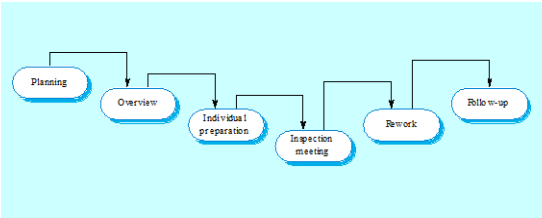
Program inspections

- Formalized 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 uninitialized variable) or non-compliance with standards.

Inspection pre-conditions

- A precise specification **should** be available.
- Team members must be familiar with the organization standards.
- Syntactically correct code or other system representations must be available.
- An error checklist should be prepared.

The inspection process



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.
- Modifications are made to repair discovered errors.
- Re-inspection may or may not be required.

Inspection roles (formal)

Author or owner	The programmer or designer responsible for producing the program or document. Responsible for fixing defects discovered during the inspection process.
Inspector	Finds errors, omissions and inconsistencies in programs and documents. May also identify broader issues that are outside the scope of the inspection team.
Reader	Presents the code or document at an inspection meeting.
Scribe	Records the results of the inspection meeting.
Chairman or moderator	Manages the process and facilitates the inspection. Reports process results to the Chief moderator.
Chief moderator	Responsible for inspection process improvements, checklist updating, standards development etc.

Inspection checklists

- Checklist of common errors should be used to drive the inspection.
- Error checklists are programming language dependent and reflect the characteristic errors that are likely to arise in the language.
- In general, the 'weaker' the type checking, the larger the checklist.
- Examples: Initialization, constant naming, loop termination, array bounds, formatting, etc.

Inspection checks 1

Data faults	Are all program variables initialised before their values are used? Have all constants been named? 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? Is there any possibility of buffer overflow?
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? If a break is required after each case in case statements, has it been included?
Input/output faults	Are all input variables used? Are all output variables assigned a value before they are output? Can unexpected inputs cause corruption?

Inspection checks 2

Interface faults	Do all function and method 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 de-allocated after it is no longer required?
Exception management faults	Have all possible error conditions been taken into account?

Inspection rate

- 500 statements/hour during overview.
- 125 source statement/hour during individual preparation.
- 90-125 statements/hour can be inspected.
- Inspection is therefore an expensive process.

Automated static analysis

- Static analyzers are software tools for source text processing. (example: Lint)
- They parse the program text and try to discover potentially erroneous conditions and bring these to the attention of the V & V team.
- They are very effective as an aid to inspections - they are a supplement to but not a replacement for inspections.
