Software Verification and Validation (V&V)

Some materials adapted from Software Engineering, 8th Ed. by Ian Sommerville

Topics covered

- Software verification and validation
- Software testing
- Software inspections
- Automated static analysis

Verification vs Validation

- Verification
  - “Are we building the product right”
  - Making sure that software conforms to its specification
- Validation
  - “Are we building the right product”
  - Making sure software does what users really need

V & V process

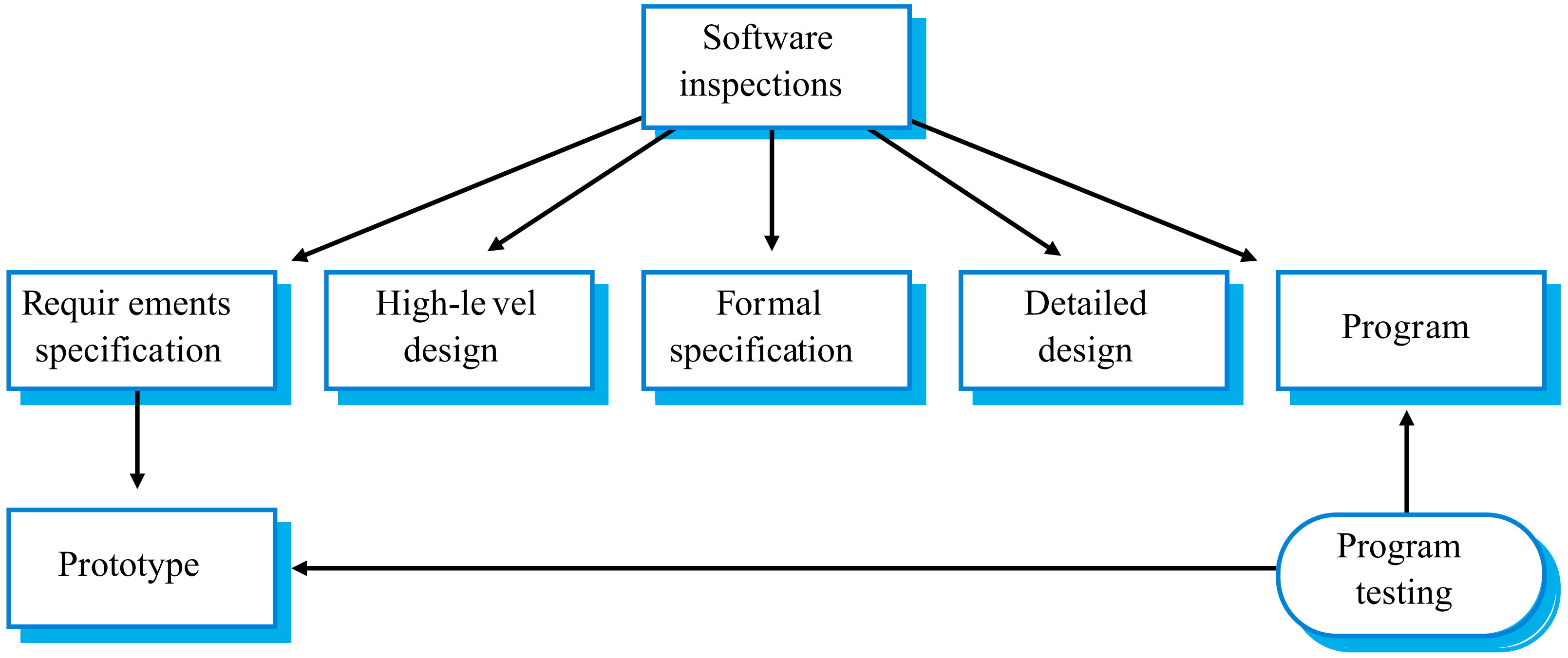
- A whole life-cycle process
  - V & V must be applied at each stage in the software process
    - Planning, requirements engineering, analysis and design, implementation, etc.
- Principal objectives of V&V process
  - Discovery of defects in software
  - Assessment of whether software is useful and useable
  - Establishing confidence in software

Static and dynamic V&V

- Software inspections (static V&V)
  - Concerned with analysis of static software representation to discover problems
  - E.g., requirements review, code review, etc.

- Software testing (dynamic V&V)
  - Concerned with exercising and observing software behavior
  - Involving running software with test data

Static and dynamic V&V



Topics covered

- Software verification and validation
- ➡ Software testing
- Software inspections
- Automated static analysis

Software testing

- Checking software to detect defects and establish confidence in software
- Can reveal the presence of defects, NOT their absence
- Used in conjunction with static verification to provide full V&V coverage

Testing vs. debugging

- Testing – establishing the existence of defects in a program
- Debugging – locating and repairing these errors

- Debugging is mainly done by developers
- Testing is done by developers, testers, customers, etc.

Topics covered

- Software verification and validation
- Software testing
- Software inspections
- Automated static analysis

Software inspections

- Involve examining work products to discover anomalies and defects
- Do not require execution of a system
- Can be applied to any representation of the system
  - requirements, plan, design, configuration data, test data, code, etc.
- An effective technique for
  - discovering errors
  - reducing problems in project
  - mitigating risks in project

Inspections and testing

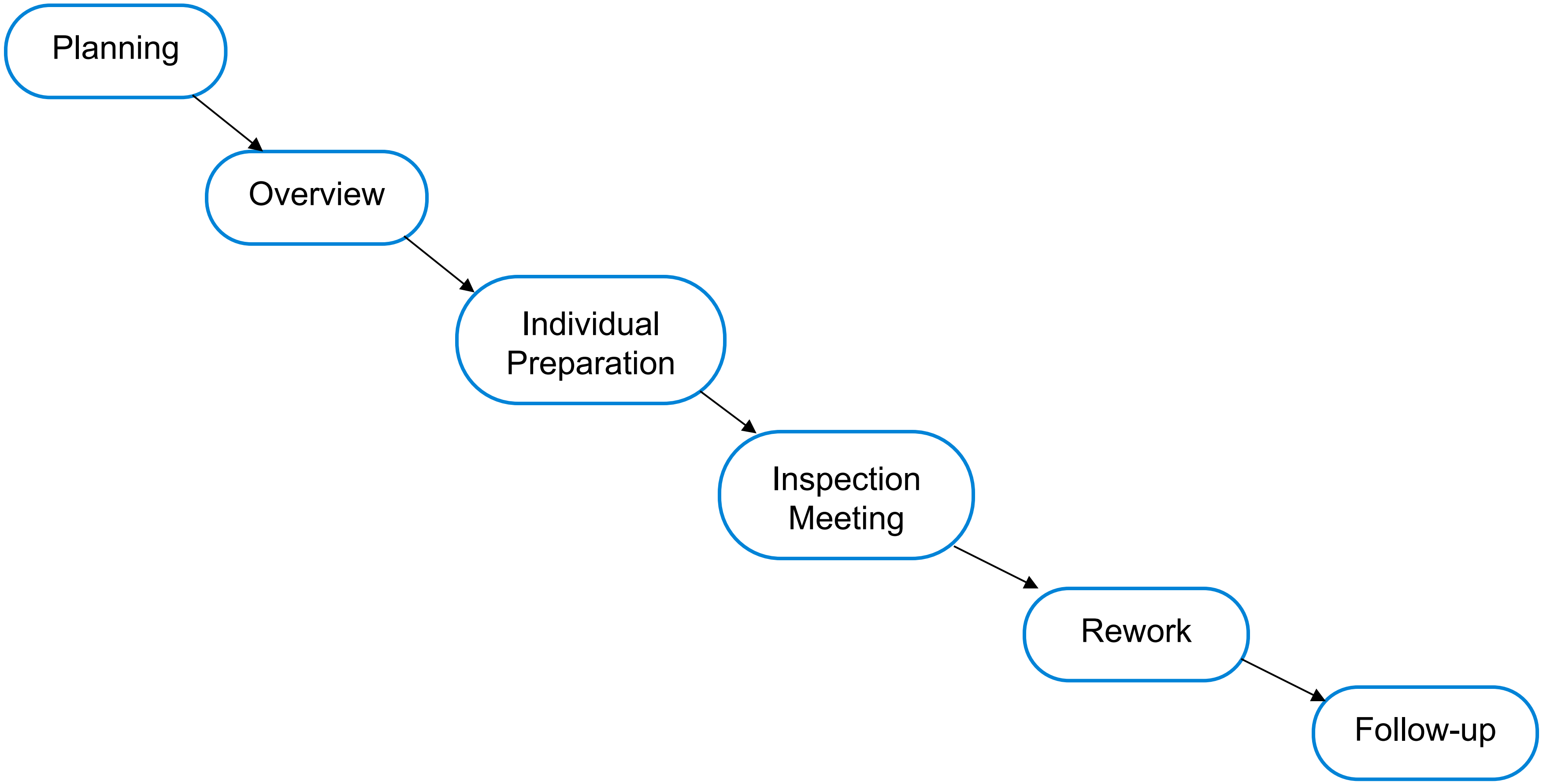
- Inspections and testing are complementary and not opposing verification techniques
- Both should be used during the V & V process
- Inspections can check conformance with a specification

Program inspections

- Formalized approach to document reviews
- Intended explicitly for defect detection

- Defects may be
  - logical errors
  - anomalies in code
  - coding standard violations
  - potential issues concerning performance, security, etc.

An inspection process



Inspection roles and responsibilities

Role	Responsibilities
Author	The programmer or designer responsible for producing the program or document, fixing defects
Inspector	Finds errors, omissions and inconsistencies in programs and documents
Reader	Presents the code or document at an inspection meeting
Scribe/recorder	Records the results of the inspection meeting
Moderator	Manages the process and facilitates the inspection. Reports process results to the Chief moderator

Inspection checklists

- Checklist of common errors should be used to drive the inspection
- Error checklists are programming language dependent

- Examples
  - 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?

Topics covered

- Software verification and validation
- Software testing
- Software inspections
- Automated static analysis

Automated static analysis

- Static analyzers are software tools for source text processing
- They parse the program text and try to discover potentially erroneous conditions
- They are very effective as an aid to inspections
- They are a supplement to but not a replacement for inspections

Stages of static analysis – 1

- Control flow analysis
  - Checks for loops with multiple exit or entry points, finds unreachable code, etc.
- Data use analysis
  - Detects uninitialized variables, variables written twice without an intervening assignment, variables which are declared but never used, etc.
- Interface analysis
  - Checks the consistency of routine and procedure declarations and their use

Stages of static analysis – 2

- Information flow analysis
  - Identifies the dependencies of output variables
  - Does not detect anomalies itself but highlights information for code inspection or review
- Path analysis
  - Identifies paths through the program and sets out the statements executed in that path
  - Potentially useful in the review process
- Both these stages generate vast amounts of information
  - They must be used with care

Use of static analysis

- Particularly valuable when a language such as C is used
- Such language has weak typing and hence many errors are undetected by the compiler
- Less cost-effective for languages, e.g., Java, C# that have strong type checking
  - can therefore detect many errors during compilation

Key points

- Verification and validation are not the same thing
  - Verification shows conformance with specification
  - Validation shows that the program meets the customer's needs
- Static verification techniques involve examination and analysis of the program for error detection

Key points

- Program inspections are very effective in discovering errors
- Program code in inspections is systematically checked by a small team to locate software faults
- Static analysis tools can discover program anomalies