

Software Engineering

Verification and Validation

Based on Software Engineering, 7th Edition by Ian Sommerville

Stan Kurkovsky

Objectives

- To introduce software verification and validation and to discuss the distinction between them
- To describe the program inspection process and its role in V & V
- To explain static analysis as a verification technique
- To describe the Cleanroom software development process

Stan Kurkovsky

Verification and validation

- **Verification:** "Are we building the product right?"
 - The software should conform to its specification.
- **Validation:** "Are we building the right product?"
 - The software should do what the user really requires.
- It is a whole life-cycle process - V & V must be applied at each stage in the software process.
- Has two principal objectives
 - The **discovery** of defects in a system;
 - The **assessment** of whether or not the system is useful and useable in an operational situation.
- Verification and validation should establish confidence that the software is fit for purpose.
- This does NOT mean completely free of defects.
- Rather, it must be good enough for its intended use and the type of use will determine the degree of confidence that is needed.

Stan Kurkovsky

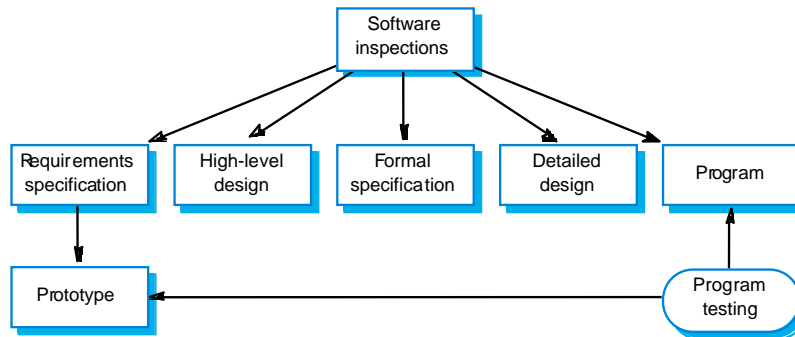
V & V confidence

- Depends on system's purpose, user expectations and marketing environment
- **Software function**
 - The level of confidence depends on how critical the software is to an organisation.
- **User expectations**
 - Users may have low expectations of certain kinds of software.
- **Marketing environment**
 - Getting a product to market early may be more important than finding defects in the program.

Stan Kurkovsky

Static and dynamic verification

- **Software inspections.** Concerned with analysis of the static system representation to discover problems (static verification)
 - May be supplemented by tool-based document and code analysis
- **Software testing.** Concerned with exercising and observing product behaviour (dynamic verification)
 - The system is executed with test data and its operational behaviour is observed



Stan Kurkovsky

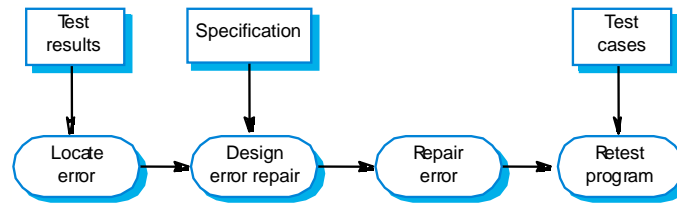
Program testing

- Can reveal the presence of errors NOT their absence.
- The only validation technique for non-functional requirements as the software has to be executed to see how it behaves.
- Should be used in conjunction with static verification to provide full V&V coverage.
- **Defect testing**
 - Tests designed to discover system defects.
 - A successful defect test is one which reveals the presence of defects in a system.
- **Validation testing**
 - Intended to show that the software meets its requirements.
 - A successful test is one that shows that a requirements has been properly implemented.

Stan Kurkovsky

Testing and debugging

- Defect testing and debugging are distinct processes.
- Verification and validation is concerned with establishing the existence of defects in a program.
- Debugging is concerned with locating and repairing these errors.
- Debugging involves formulating a hypothesis about program behaviour then testing these hypotheses to find the system error.

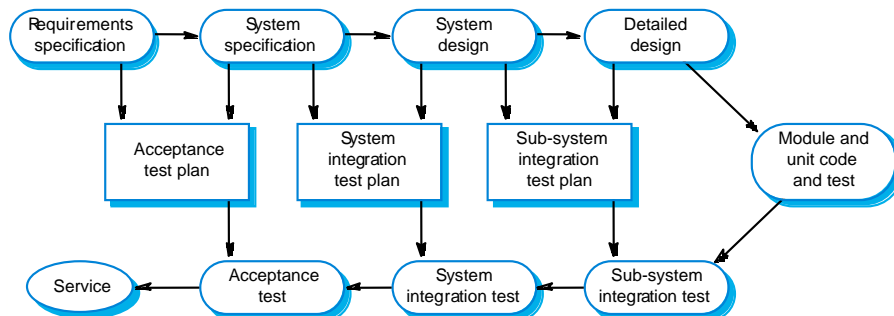


Stan Kurkovsky

V & V planning

- Careful planning is required to get the most out of testing and inspection processes.
- Planning should start early in the development process.
- The plan should identify the balance between static verification and testing.
- Test planning is about defining standards for the testing process rather than describing product tests.

The V-model of development



Stan Kurkovsky

The software test plan

- **The testing process**
 - A description of the major phases of the testing process. These might be as described earlier in this chapter.
- **Requirements traceability**
 - Users are most interested in the system meeting its requirements and testing should be planned so that all requirements are individually tested.
- **Tested items**
 - The products of the software process that are to be tested should be specified.
- **Testing schedule**
 - An overall testing schedule and resource allocation for this schedule. This, obviously, is linked to the more general project development schedule.
- **Test recording procedures**
 - It is not enough simply to run tests. The results of the tests must be systematically recorded. It must be possible to audit the testing process to check that it been carried out correctly.
- **Hardware and software requirements**
 - This section should set out software tools required and estimated hardware utilisation.
- **Constraints**
 - Constraints affecting the testing process such as staff shortages should be anticipated in this section.

Stan Kurkovsky

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.

Stan Kurkovsky

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

- 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 uninitialized variable) or non-compliance with standards.

Stan Kurkovsky

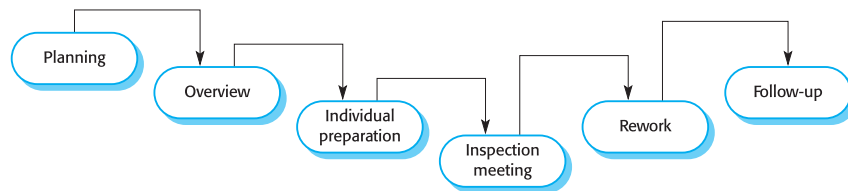
Inspection pre-conditions

- A precise specification must be available.
- Team members must be familiar with the organisation standards.
- Syntactically correct code or other system representations must be available.
- An error checklist should be prepared.
- Management must accept that inspection will increase costs early in the software process.
- Management should not use inspections for staff appraisal i.e. finding out who makes mistakes.

Stan Kurkovsky

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.



Stan Kurkovsky

Inspection checks

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?

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?

Stan Kurkovsky

Automated static analysis

- Static analysers are software tools for source text processing.
- 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.

- Data faults
 - Variables used before initialisation
 - Variables declared but never used
 - Variables assigned twice but never used between assignments
 - Possible array bound violations
 - Undeclared variables
- Control faults
 - Unreachable code
 - Unconditional branches into loops
- Input/output faults
 - Variables output twice with no intervening assignment
- Interface faults
 - Parameter type mismatches
 - Parameter number mismatches
 - Non-usage of the results of functions
 - Uncalled functions and procedures
- Storage management faults
 - Unassigned pointers
 - Pointer arithmetic

Stan Kurkovsky

Stages of static analysis

- **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
- **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. Again, potentially useful in the review process
- The last two stages generate vast amounts of information. They must be used with care.

Stan Kurkovsky

Verification and formal methods

- Formal methods can be used when a mathematical specification of the system is produced.
- They are the ultimate static verification technique.
- They involve detailed mathematical analysis of the specification and may develop formal arguments that a program conforms to its mathematical specification.

Pros

- Producing a mathematical specification requires a detailed analysis of the requirements and this is likely to uncover errors.
- They can detect implementation errors before testing when the program is analyzed alongside the specification.

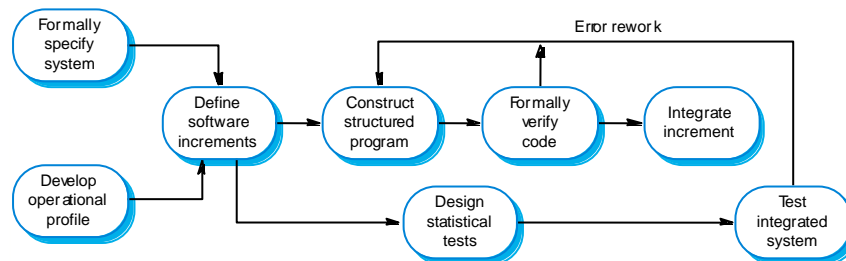
Cons

- Require specialized notations that cannot be understood by domain experts.
- Very expensive to develop a specification and even more expensive to show that a program meets that specification.
- It may be possible to reach the same level of confidence in a program more cheaply using other V & V techniques.

Stan Kurkovsky

Cleanroom software development

- The name is derived from the 'Cleanroom' process in semiconductor fabrication. The philosophy is defect avoidance rather than defect removal.
- This software development process is based on:
 - Incremental development;
 - Formal specification;
 - Static verification using correctness arguments;
 - Statistical testing to determine program reliability.



Stan Kurkovsky

Cleanroom process characteristics

- Formal specification using a state transition model.
- Incremental development where the customer prioritises increments.
- Structured programming - limited control and abstraction constructs are used in the program.
- Static verification using rigorous inspections.
- Statistical testing of the system

Formal specification and inspections

- The state based model is a system specification and the inspection process checks the program against this model.
- The programming approach is defined so that the correspondence between the model and the system is clear.
- Mathematical arguments (not proofs) are used to increase confidence in the inspection process.

Stan Kurkovsky

Cleanroom process teams and process evaluation

- **Specification team.** Responsible for developing and maintaining the system specification.
- **Development team.** Responsible for developing and verifying the software. The software is NOT executed or even compiled during this process.
- **Certification team.** Responsible for developing a set of statistical tests to exercise the software after development. Reliability growth models used to determine when reliability is acceptable.
- The results of using the Cleanroom process have been very impressive with few discovered faults in delivered systems.
- Independent assessment shows that the process is no more expensive than other approaches.
- There were fewer errors than in a 'traditional' development process.
- However, the process is not widely used. It is not clear how this approach can be transferred to an environment with less skilled or less motivated software engineers.

Stan Kurkovsky

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.
- Test plans should be drawn up to guide the testing process.
- Static verification techniques involve examination and analysis of the program for error detection.
- 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 which may be an indication of faults in the code.
- The Cleanroom development process depends on incremental development, static verification and statistical testing.