XVII. Verification and Validation

Objective

- 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

Verification vs. 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.

The V&V process

- 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.

V&V goals

- **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.**
- **Confidence is certainly subjective and depends on many factors**

V&V Confidence

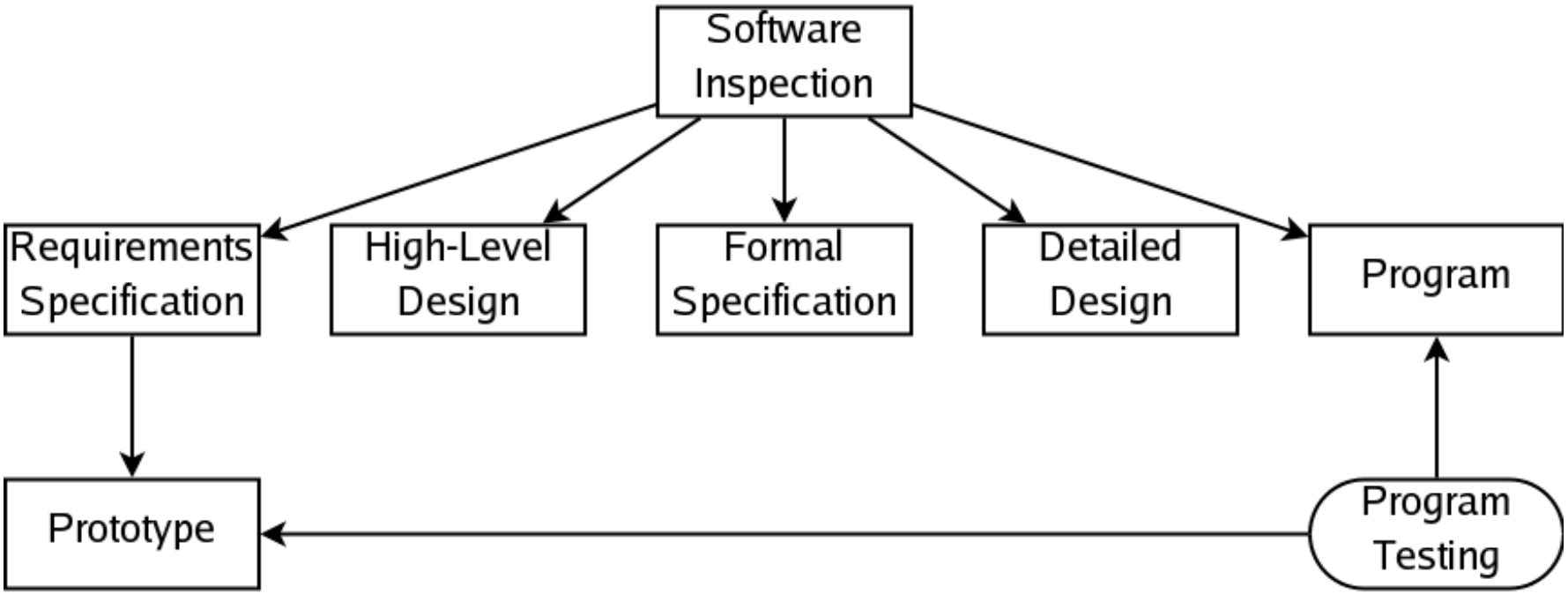
- Depends on system's purpose, user expectations and marketing environment
- **Software function**
  - The level of confidence depends on how critical the software is to an organisation.
- **User expectations**
  - Users may have low expectations of certain kinds of software.
- **Marketing environment**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

Static vs Dynamic Verification

- **Software inspections**. Concerned with analysis of the static system representation to discover problems (static verification)
  - May be supplement 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

Static and Dynamic V&V

- Inspection and Testing play complementary roles:



- Testing requires the availability of an executable version
- **Testing can reveal the presence of errors not their absence**

Types of Testing

Defect testing

- Tests designed to discover system defects.
- A successful defect test is one which reveals the presence of defects in a system.
- Covered in Chapter 23

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.
- In general requires the definition of operational profiles and the evaluation of reliability

Testing vs. 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

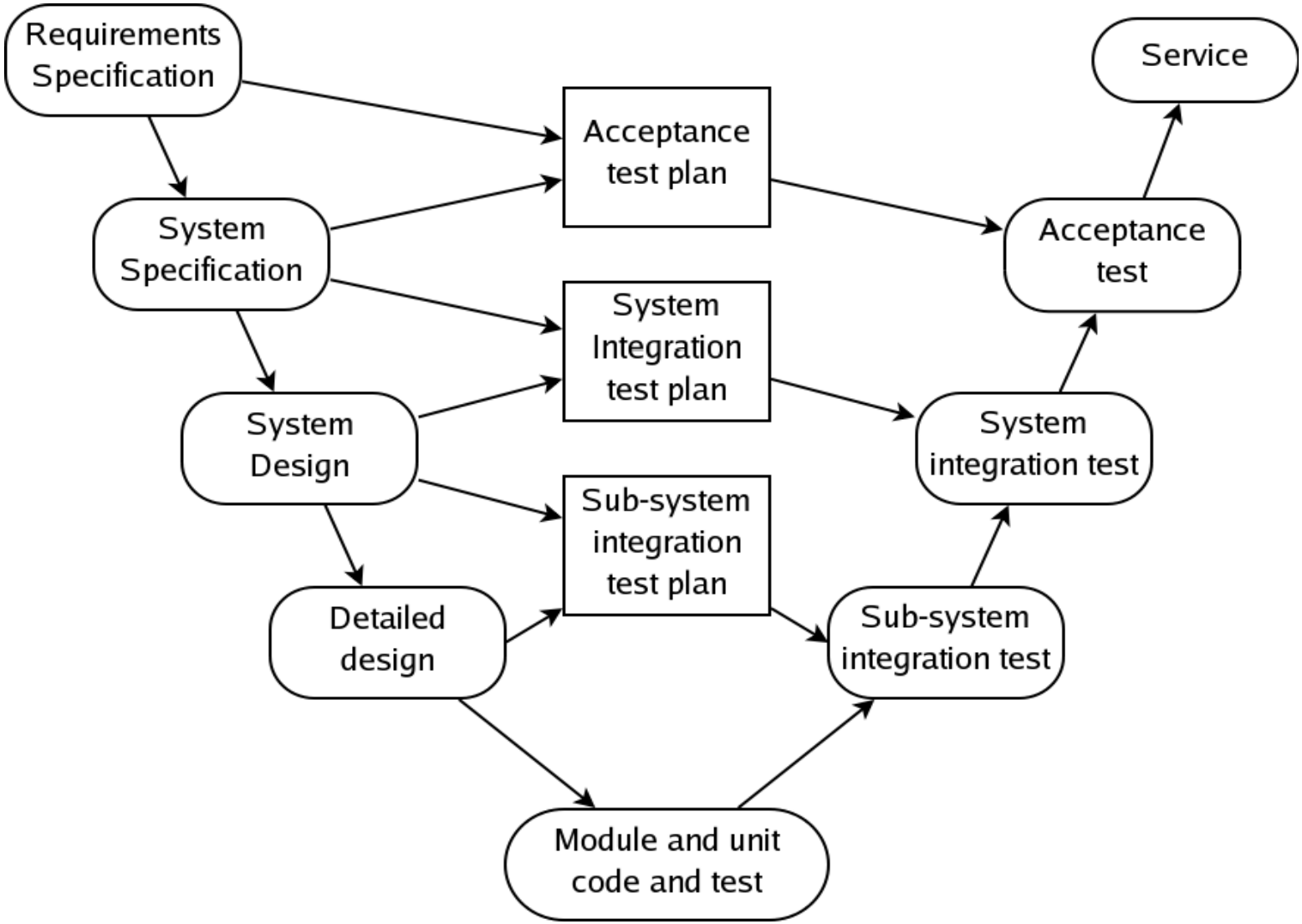
- Debugging requires to collect information on the execution
- **Interactive debugging** permits to follow software dynamics

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

- Test plan should be derived from system specification and design



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.

The Software Test Plan

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.

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 and Testing

- **Many different defects may be discovered in a single inspection**.

  In testing, one defect, may mask another so several executions are required.
- Incomplete version of a system can be inspected
- The reuse domain and programming knowledge so reviewers are likely to have seen the types of error that commonly arise
- 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.

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.

Inspection Process

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

Inspection roles

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.

Check List

- Inspection it is made easier by the use of check-list
  - Data faults
  - Control faults
  - input/output faults
  - Interface faults
  - Storage management faults
  - Exception management faults

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.

- Eclipse provide some automated static analysis feature
- More effective for weak typed languages such as C
- Java and C# reduce error sources – the compiler can already check many sources

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 and Cons

- 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 analysed alongside the specification.

- Require specialised 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.