Model-Based Testing

(DIT848 / DAT260)

Spring 2015

Lecture 1

Overview of Verification and Validation

Gerardo Schneider
gerardo@cse.gu.se
Dept. of Computer Science and Engineering
Chalmers | University of Gothenburg

Some slides based on material by Magnus Björk, Thomas Arts and Ian Somerville

Lecture 1

- Introduce software verification and validation and discuss the distinction between them

- Introduce link between development and test

Lots of new words, putting them into context

Discuss: What is SW quality?

* Downloaded from youtube

Quality aspects considered in this course

High priority

- Correctness:
  - The program should fulfill its specification
  - The program should not malfunction (crash, etc)

Lower priority

- Suitability
- Usefulness
- Code maintainability / standards conformance
- Document quality

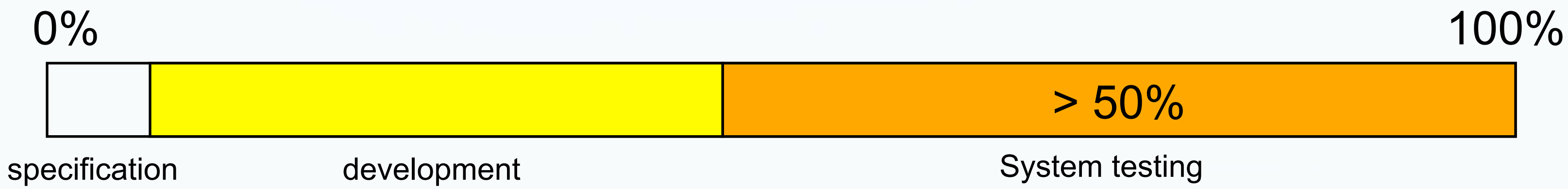
Motivation

Product development costs

How much do you think testing “costs”?

Motivation

Product development costs (Sommerville)



The more mature innovations get, the more important is their quality

Example: GPS receiver

"Software quality" is getting a competitive distinction

The company being able to test better for less money gets the market

Bugs are serious

Ariane 5 flight 501

- Error in a code converting 64-bit floating-point numbers into 16-bit signed integer: It triggered an overflow condition
- Rocket disintegrate 40 seconds after launch
- Price: ~USD 370M in equipment

- Therac-25 Radiation therapy machine
  - Possible to configure the Therac-25 so the electron beam would fire in high-power mode but with the metal X-ray target out of position
  - Source of error: a "race condition"
  - Price: 5 people killed by massive overdoses

Verification & Validation

- Verification

"Are we building the product right"

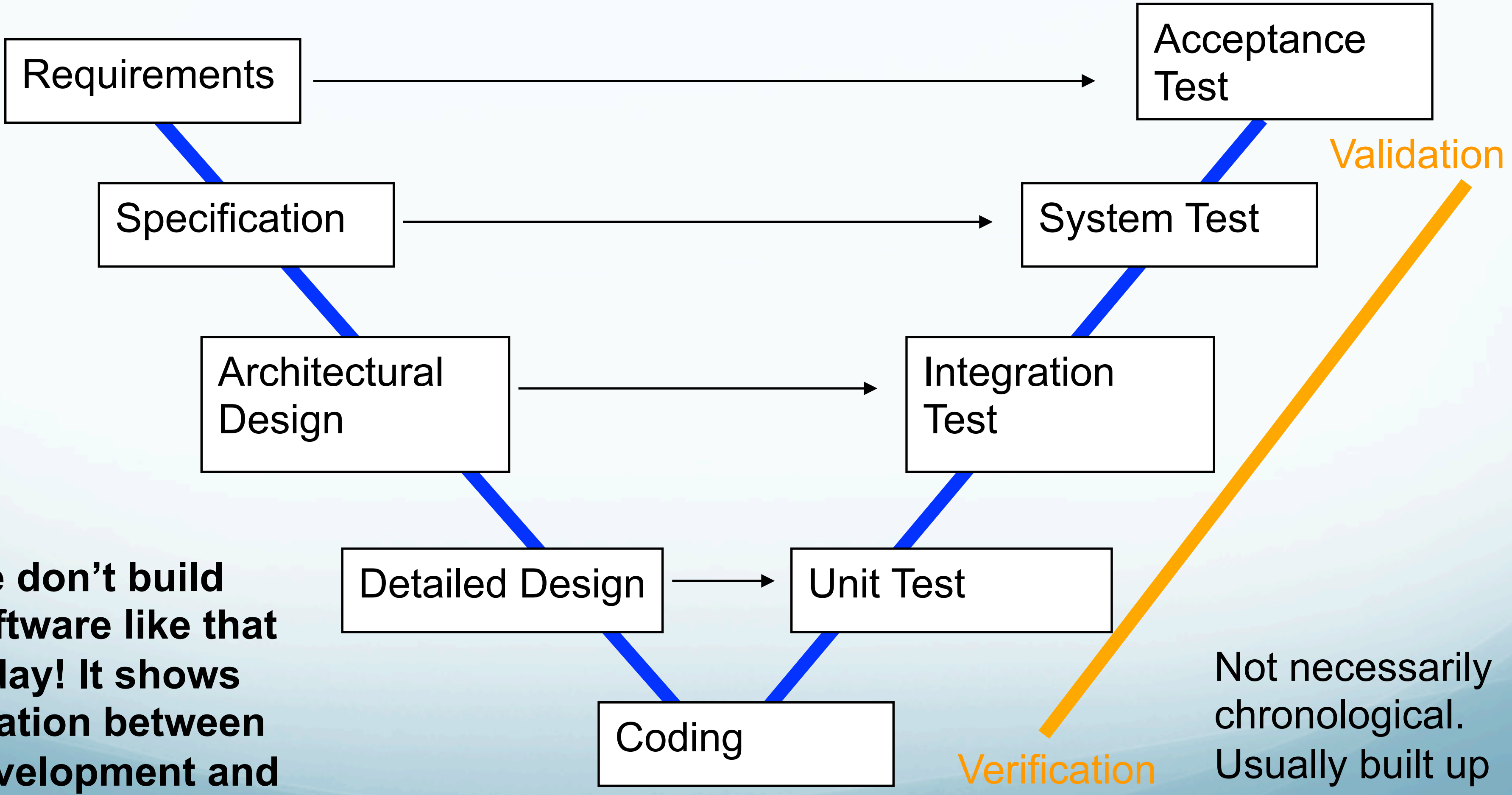
The software should conform to its specification

- Validation

"Are we building the right product"

The software should do what the user really requires

V model [cf Spillner 2000]



We don’t build software like that today! It shows relation between development and verification

Not necessarily chronological. Usually built up iteratively.

How does it work in practice?

- This is what we will see in this course…
- Remember that the V-model is useful to show how development and test are related conceptually
  - In practice, different ways to organize/perform testing
- We will see “traditional” ways of performing testing
- And obviously Model-Based Testing (MBT)

Dynamic and static verification

- *DYNAMIC*
  – *Software testing* & Runtime verification
  - Concerned with exercising and observing product behaviour
  - The system is executed with test data and its operational behaviour is observed

- *STATIC*
  – *Software inspections* & Other model-based techniques (besides MBT)
  - Concerned with analysis of the static system (representation) to discover problems
  - May be supplemented by tool-based document and analysis

Error, defect, failure…

- *Error*

A human action that produces an incorrect result

  - Mistakes in syntax, wrong invocation, wrong initialization of variables, …

- *Defect* *(or bug, or fault)*

A flaw in a component/software that might cause the system to fail to perform its required function
If encountered during execution: might cause a *failure*

  - Incorrect statement or data definition

- *Failure*

Deviation of the component/software from its expected result

  - The program crashes, the wrong result is obtained

According to the "*International Software Testing Qualification Board*" (ISTQB)
Warning: Not everybody agrees with the above distinction!

The V&V process

- It's a whole life-cycle process
  - V&V must be applied at each stage in the software process
  - So, V&V and development processes depend on each other

- 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 process

Goals

- Verification and validation should increase confidence on that the software fits the intended purpose

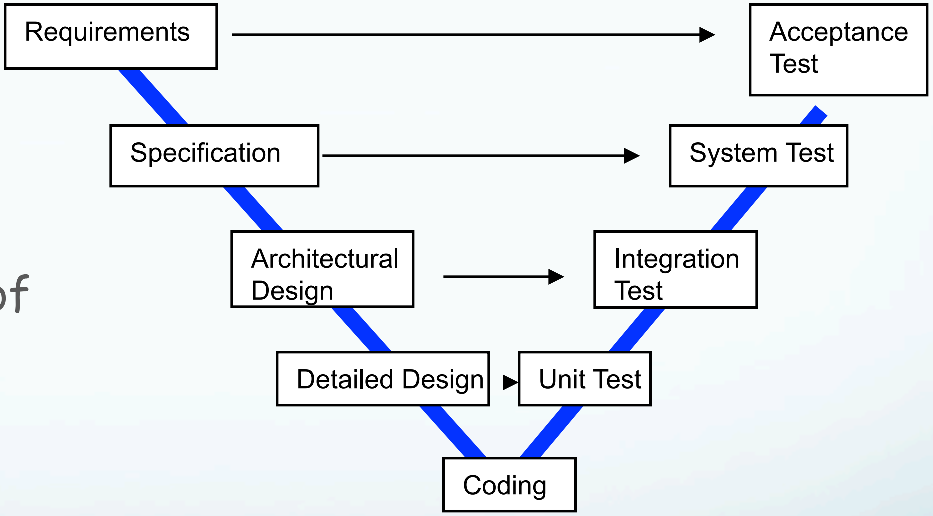
- This does **NOT** mean completely free of defects

- Rather, it must be good enough for its intended use and the type of use will determine the degree of confidence that is needed

V&V process

Confidence on Sw correctness depends on

- Software function
  - How critical the software is to an organization
- 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
- Patchability
  - Can sold units be upgraded easily?



Requirements → Acceptance Test

Specification → System Test

Architectural Design → Integration Test

Detailed Design → Unit Test

Coding

Discussion
Software Testing in Automobiles

Discuss software in the car

Discuss for several software components:

- How critical they are
- What the users expect
- How the marketing environment looks like
- Whether upgrades are feasible

Offset crash test at 64km/h

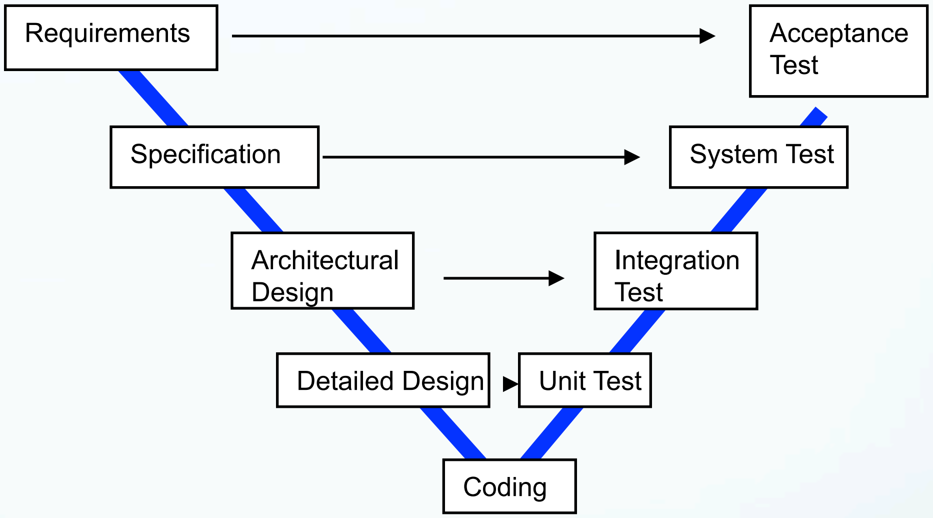
V&V planning

- Careful planning is required to get the most out of static and dynamic verification
- **Planning should start early in the development process**
- The plan should identify the balance between dynamic and static “verification” (between testing and inspection)
- V&V planning is about defining standards for the V&V process, rather than describing product tests
- **The more critical the system, the more effort should be devoted to *static verification***

V&V planning

Plan V&V process

- Which activities?
- Which results for each activity?
- Who performs activity?



V-model helps to connect test activities to development activities

Each development activity corresponds to a test level

Test levels

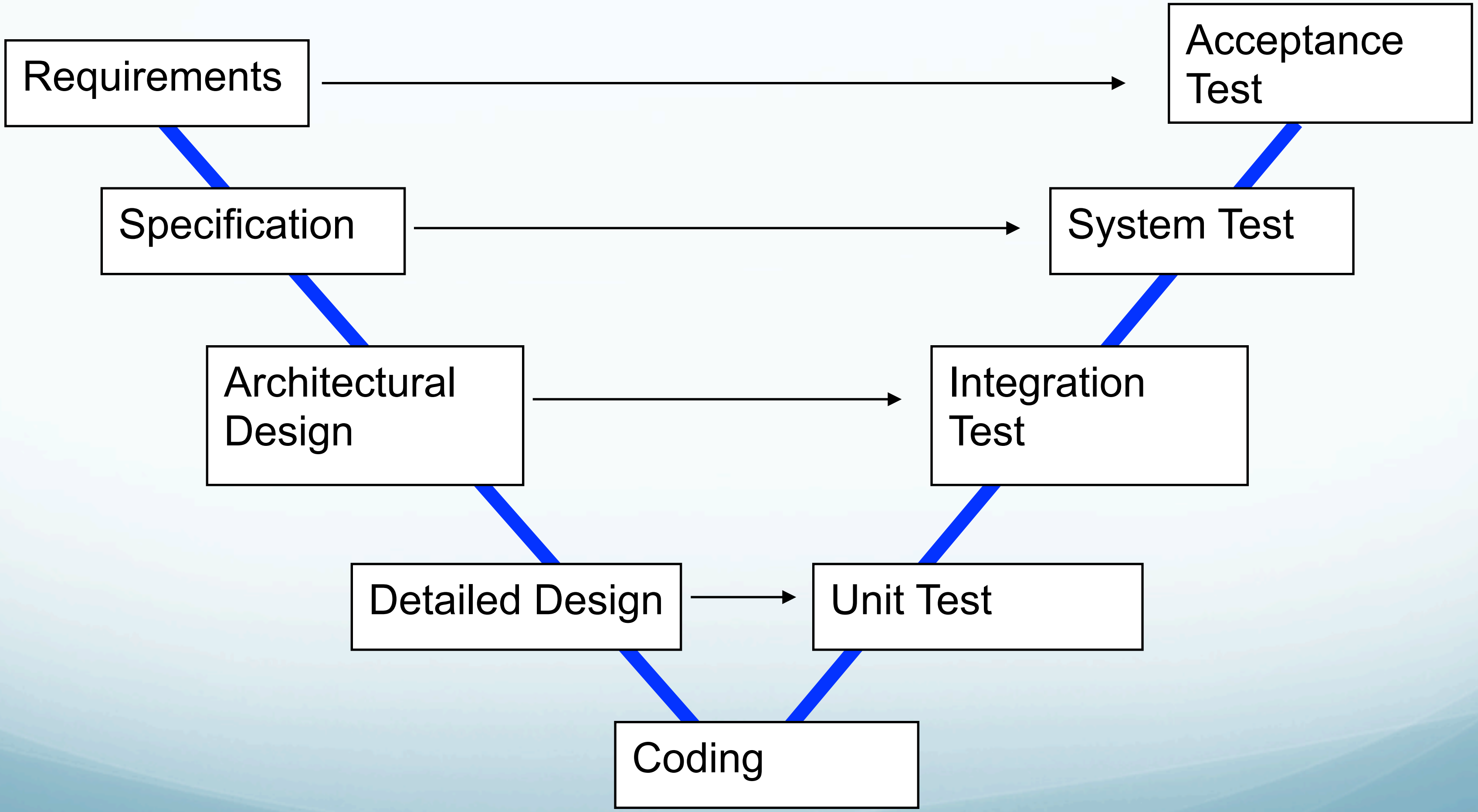
Test level: A group of test activities that are organized and managed together

A test level is linked to responsibilities in a project

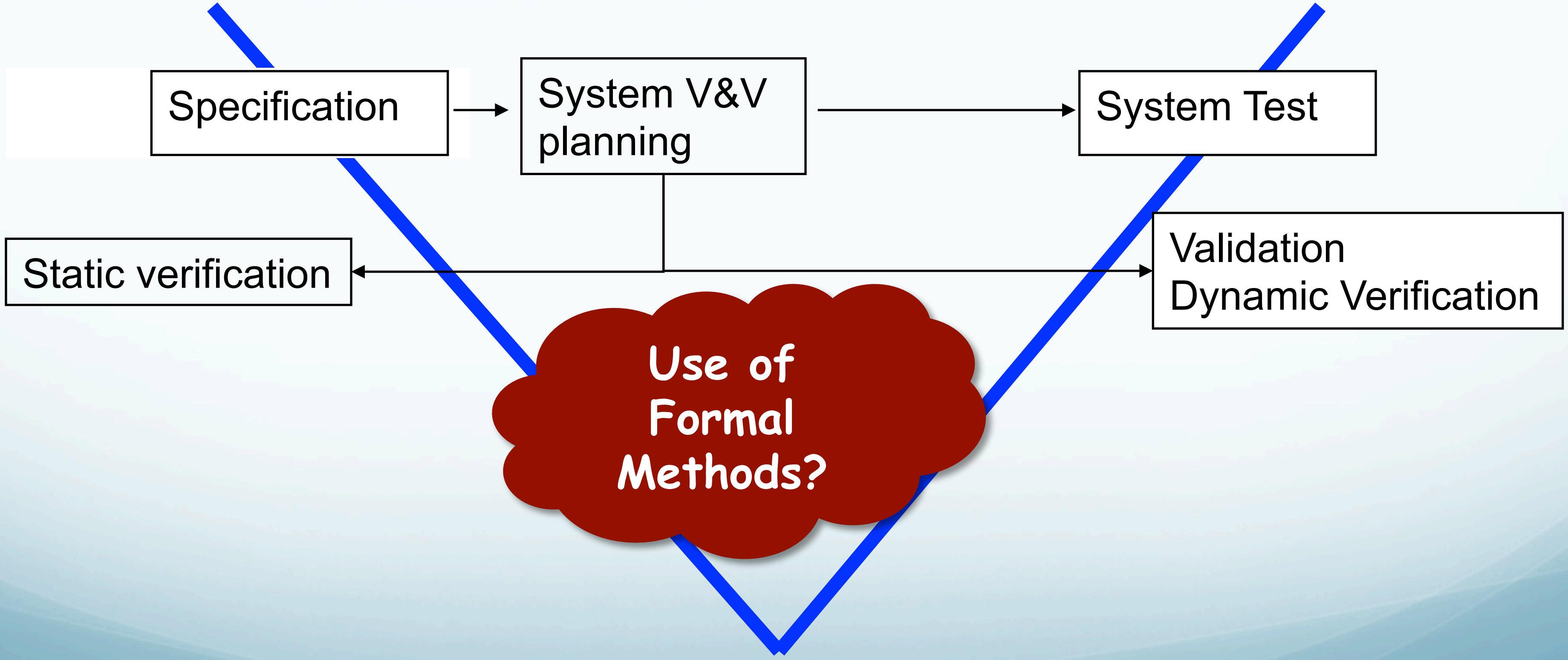
For each level, it is important to test what was not possible to verify or validate on lower levels

Different methods and techniques may apply to each level

Verification and Validation



Verification and Validation



Specification

System V&V planning

System Test

Static verification

Validation
Dynamic Verification

Use of Formal Methods?

Dynamic verification

- Testing can reveal the presence of errors NOT their absence (Dijkstra 1960's)
- The "only" validation technique for non-functional requirements as the software has to be executed to see how it behaves
  - Non-exhaustive
- Should be used in conjunction with static verification to approximate a full V&V coverage

Types of testing (one possible classification)

- 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
  - *Quality assurance process* carried out before the software is ready for release
  - Intended to show that the software meets the requirements given by the user
    - Acceptance by the end user
  - A successful test is one that shows that requirements have been properly implemented

Testing and debugging

- Defect testing and debugging are distinct processes

- Testing 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

Costs of debugging are often included in costs for Software testing

Software inspections

Software inspection is a manual static verification method

- It involves people/tools examining the source representation with the aim of discovering anomalies and defects
- Inspections can take place on all development levels, no matter the formality of the sources
- Inspections do 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.)
- Shown to be an effective technique for discovering program errors

XP: pair programming

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
- Incomplete versions of a system can be inspected without extra cost
- You can look for inefficiencies, poor programming style, etc

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 (partial) conformance with a specification but **not** conformance with the customer's real requirements
- Inspections **cannot** check non-functional characteristics such as performance, usability, etc.
- But inspections can find other non-functional characteristics such as standards compliance of code

Verification and formal methods

- Formal methods can be used when a mathematical specification of the system is known

- They are the *ultimate* verification technique

- They involve detailed mathematical analysis of the specification and may develop formal arguments that a program conforms to its mathematical specification

Typical testing methods on each level

- Unit tests:
  - Each programmer required to write unit tests for own code, organized in automatically executable test suites
  - Automatic static verification (lint/splint-like)
  - Manual code inspections
- Integration tests:
  - Write test cases that monitor how modules interact
  - Some manual code inspections
- System tests:
  - Scripted test suite (especially if text based program)
  - Manual tests – trying to break the system
- Acceptance tests:
  - Customer manually tests software

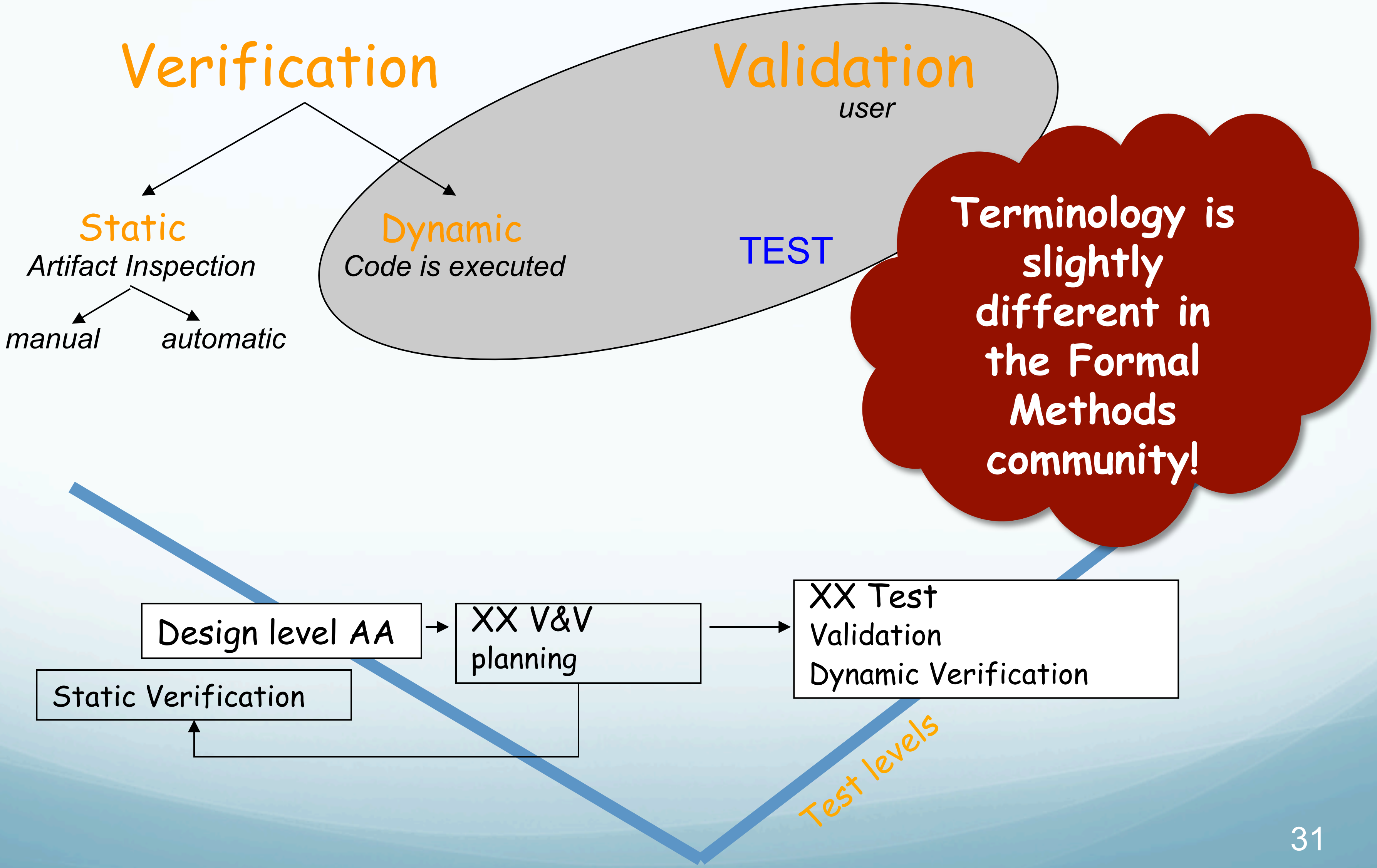
Model-Based testing (automatic test extraction from a model) not specifically associated with a level – need of a model!

Conclusions

- Verification and validation are not the same thing
  - Verification shows conformance with specification;
  - Validation shows that the program meets the customer's needs

- V&V plans should be drawn up to guide the V&V process (part of the V&V plan is a test plan)
- Each design activity has a corresponding V&V activity
- Static verification techniques involve examination and analysis for error detection (among others)
- Dynamic verification implies "running" the code

Terminology



Verification

Validation

user

Static

Artifact Inspection

manual

automatic

Dynamic

Code is executed

TEST

Terminology is slightly different in the Formal Methods community!

Design level AA

XX V&V planning

XX Test

Validation

Dynamic Verification

Static Verification

Test levels

Literature

- Jorgensen, *Software Testing: A Craftsman's Approach.*
  - Chapter 1
- Ian Sommerville, *Software Engineering*
  - Chapter 22.1-2 + 23.1-2, Edition 7 or 8

Another software bug…

- Posted on YoutTube on August 15, 2009
  - Fixed by Apple few months later